ERDC/CRREL TR-10-12

Cold Regions Research and Engineering Laboratory

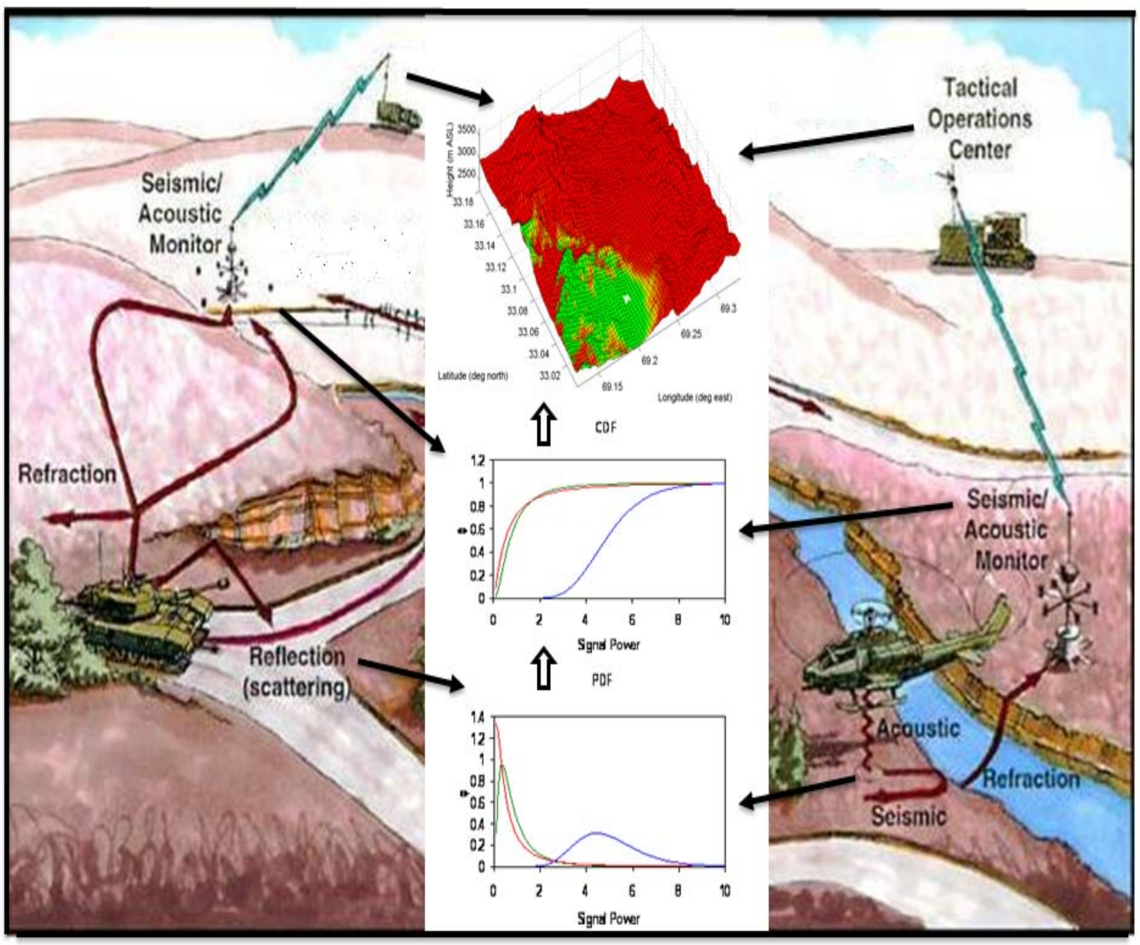
US Army Corps of Engineers®
Engineer Research and Development Center

Geospatial Research and Engineering

Probability and Statistics in Sensor Performance Modeling

Kenneth K. Yamamoto, D. Keith Wilson, and Chris L. Pettit

December 2010



Approved for public release; distribution is unlimited.

Geospatial Research and Engineering

ERDC/CRREL TR-10-12
December 2010

Probability and Statistics in Sensor Performance Modeling

Kenneth K. Yamamoto, D. Keith Wilson, and Chris L. Pettit

Cold Regions Research and Engineering Laboratory
U.S. Army Engineer Research and Development Center
72 Lyme Road
Hanover, NH 03755-1290

Final report

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under AT42 GEOINT Exploitation in Man-Made Environments: Nations to Insurgents (GEMENI)

Abstract: Signals from many military targets of interest are often strongly randomized, due to the irregular mechanisms by which the signals are generated and propagated. In particular, complicated and dynamic terrestrial/atmospheric environments (with man-made objects, vegetation, and turbulence) randomize signals through random atmospheric and terrestrial processes affecting the propagation. Signals may also be considered random due to uncertainties in the knowledge of the propagation environment and target-sensor geometry. Predictions of sensor performance and recommendations of sensor types and placements derived from them, thus, should account for the random nature of the sensed signals. This report discusses software-modeling approaches for characterizing signals subject to random generation and propagation mechanisms. By representing signals with random variables, they are manipulated statistically to make probabilistic predictions of sensor performance. Both the theory and implementation in a general, object-oriented software design for battlefield signal transmission and sensing is explained. The Java-language software program is called Environmental Awareness for Sensor and Emitter Employment. Some important numerical issues in the implementation are also discussed.

Contents

Acronyms iv

Preface v

1 Introduction 1

2 Statistical Modeling of Signals 3

Signature data features 3
Probabilistic modeling of features 3
Statistical analysis for measuring sensor performance 4

3 Software Implementation 7

Signal model objects 7
Signal model inheritance relationships 8
Representing a specific feature via a signal model 9

4 Sums of Random Variables 11

A very simple scheme 11
Using the Laplace approximation 14
The convolution approach 14
Central limit theorem 15

5 Numerical Issues 17

Computing the quantile using the bisection method 17
Computing the cdf using Gauss-Legendre quadrature on the pdf 18
Numerical stability and well-posed problems 20
Illustrative examples of numerically stable statistical computations 22

6 Conclusions 28

References 29

Appendix: Floating-Point Implementations of Statistical Formulae 31

Floating point 31
Stable statistical implementations 33
Gaussian distribution 33
Lognormal distribution 36
Exponential distribution 40
Gamma distribution 42
$X \rightarrow X^2$ transformed Rice-Nakagami distribution 49

Report Documentation Page 55

Acronyms

ccdf	complementary cumulative distribution function
cdf	cumulative distribution function
DST	decision-support tool
EASEE	Environmental Awareness of Sensor and Emitter Employment
FFT	fast Fourier transform
IEEE	Institute of Electrical and Electronics Engineers
NaN	not a number (value)
pdf	probability density function

Preface

This study was conducted for the U.S. Army Corps of Engineers. Funding was provided by the Engineer Research and Development Center (ERDC) Geospatial Intelligence (GEOINT) program, Exploitation in Man-made Environments: Nations to Insurgents (GEMENI).

The work was performed by the Signature Physics Branch (RR-D) of the Research and Engineering Division (RR) at ERDC's Cold Regions Research and Engineering Laboratory (ERDC-CRREL). The principal investigator was Dr. D. Keith Wilson (RR-D). The authors thank Dr. George G. Koenig of ERDC-CRREL's Terrestrial and Cryospheric Sciences Branch (RR-G) and Mr. Michael Parker of ERDC-CRREL's Force Projection and Sustainment Branch (RR-H) for providing helpful comments on a draft version of this report.

At the time of publication, Dr. Lindamae Peck was Chief, RR-D; Dr. Justin B. Berman was Chief, RR; and Mr. Dale R. Hill was the Acting Technical Director for Geospatial Research and Engineering. The Deputy Director of ERDC-CRREL was Dr. Lance D. Hansen, and the Director was Dr. Robert E. Davis.

COL Kevin J. Wilson was the Commander and Executive Director of ERDC, and Dr. Jeffery P. Holland was the Director.

1 Introduction

A variety of modern-day Army missions rely on effective sensing capabilities that provide intelligence on the adversary and to protect friendly forces from enemy detection. Sensors that are stationary (e.g., microphones, geophones, and ground-based radars) and moving (e.g., cameras on unattended aerial vehicles and ground vehicles) assist operations such as persistent surveillance of small, forward-operating bases and rapid covert troop maneuvers in the air and on the ground. When advantageous, sensing is often performed in multiple signal modalities including visible, infrared, acoustic, seismic, radiofrequency, chemical, and biological.

Yet, despite the increasing assortment of sensors available, knowledge and expertise of how to use them efficiently in particular environments and missions is quite frequently lacking (Hieb et al. 2007). Since terrain and weather effects on signals are complex and oftentimes counterintuitive, computational simulations are valuable to both facilitate quick and accurate decision making when planning the use of sensors in an actual mission as well as to guide the development of sensor technologies in general. When multiple diverse signal modalities are involved, it would be ideal to integrate them all into a single simulation environment rather than having one for each modality.

Environmental Awareness of Sensor and Emitter Employment (EASEE) is a software framework that provides a single environment for analyzing sensor performance involving many different signal modalities. This ability for multimodal signal analysis then enables EASEE to also perform higher-level data synthesis needed to answer critical sensing questions such as the sensor types and locations best suited for accomplishing mission objectives within the constraints of a particular environment. A comprehensive description of the software design and structure of EASEE is provided in Wilson et al. (2009).

In addition to reconciling models of many, differing signal modalities into a common software framework, a decision-support tool (DST) must also systematically assess the uncertainty with its predictions. Accounting for and measuring uncertainty is often overlooked in current software tools predicting sensor performance, but the reliability of a DST's recommenda-

tions is obviously key to well-informed decision making. In the complicated process of modeling environmental effects on signal transmission and sensing, many types and levels of uncertainty are involved. A thorough explanation of practical uncertainty issues is available in Wilson et al. (2008).

Generally, uncertainties may be due to incomplete and/or inaccurate knowledge (of, say, the weather for a particular time and location) or due to processes that are purely stochastic (e.g., sensitivity of wave propagation to irresolvable and highly variable environmental details). It is the latter type of uncertainty—namely stochastic/random uncertainty—with which this report is concerned. (Note that sensor performance is also affected by non-stochastic processes not addressed here.) In principle, propagation of acoustic, seismic, and electromagnetic (including visual and infrared) waves as well as the dispersion of transient gases are still sensitive to many fine-scale and even dynamic environmental variations that cannot be known or determined. Practically speaking, second-to-second variability in signal characteristics (e.g., sound level, seismic energy, concentration of a chemical agent, etc.) caused by unobservable and unpredictable environmental features (e.g., turbulent eddies, vegetation, dust particles, precipitation, small urban structures) are essentially random, so statistical models are necessary to represent signal and noise (Strohbehn 1978; Andrews and Phillips 2005). Wilson et al. (2008) outlines specific examples in greater detail. Probabilistic information about signal and noise may then be processed to compute statistical metrics on sensor performance and signal-emitter detectability.

The report begins by first explaining the premise and theory of modeling signals statistically in Chapter 2, which also describes how statistical modeling of signals lead to probabilistic predictions on sensor performance. Then, Chapter 3 explains how the statistical modeling is implemented efficiently in EASEE's general, object-oriented structure. A discussion on various approaches for summing multiple random variables is given in Chapter 4, which is required when simulating multiple target and noise signals arriving at a sensor. Some general numerical issues in programming statistical computations are illustrated in Chapter 5, while the Appendix supplies specific information on how particular statistical calculations were implemented.

2 Statistical Modeling of Signals

Signature data features

The EASEE software design (Wilson et al. 2009) operates on basic, identifying qualities of a signal called *signature data features* (or *features* for short). The features can be thought of as the most essential characteristics of signals that might identify their source (i.e., an observable that is a function of time at the source, sensor, or somewhere in between). Generally, features are extracted from raw sensor data after some low-level processing (e.g., calibrations to remove sensor response or filtering into spectral bands), typically being a conservative quantity (such as energy or mass). Examples include: sound power of a harmonic line or in a standard octave band; infrared brightness in near, shortwave, midwave, longwave, or far bands; components of an electromagnetic field vector; and concentration of a particular chemical or biological species. By using features rather than raw signals, simulations like EASEE are made not only quicker and much more efficient but also more general and nonspecific to any particular sensor system.

Probabilistic modeling of features

As discussed earlier, irregular generation mechanisms and random propagation processes may make it impractical to predict feature characteristics *deterministically*; rather, they can only be described *probabilistically*. Thus, signal and noise features may be regarded as random variables. Then, *statistical distributions* of signal features are generated, propagated, and processed.

A variety of statistical models are appropriate for representing sensor data. For example, the exponential distribution describes a single, strongly scattered signal, which often arises when acoustic or electromagnetic waves are scattered by both objects and turbulent wind. A version of the Rice-Nakagami model (specifically with a variable transformation of $X \rightarrow X^2$) may closely approximate signal power distributions of certain acoustic and seismic signals. Since the lognormal distribution models variables that can be the multiplicative product of many independent, positive random variables, atmospheric scientists often use it to describe plume sizes and frequency distributions of transient gases from turbulent processes in the air

(Baker et al. 1983; Limpert et al. 2001). Generally, signals are represented with probability density functions (pdfs) that have nonnegative support because signal power or concentration can never be negative. However, if the mean is positive and much larger than the standard deviation, pdfs with real-number support like the Gaussian may also be suitable for representing a signal. Currently, EASEE implements five continuous statistical models—namely, the Gaussian, lognormal, exponential, gamma, and the $X \to X^2$ transformed Rice-Nakagami—as well as a discrete model. (Other examples of statistical models for signals are in Burdic 1984; Wilson et. al. 2002; and Nadarajah 2008)

For consistency, all of these models are used to describe distributions of signal power (or intensity) rather than amplitude. Thus, while the untransformed Rice-Nakagami model may describe the distribution of signal amplitudes of various acoustic and seismic signals, it is necessary to find the corresponding model that would represent the distribution of the signal powers. Since signal power is related to the square of the amplitude, the correct model for the signal-power distribution is the Rice-Nakagami with a variable transformation of $X \to X^2$, which is what is coded in EASEE. (Whenever the "transformed Rice-Nakagami" is referred to in this report, the $X \to X^2$ transformed Rice-Nakagami model is meant.) Details, including the derivation of the $X \to X^2$ transformed Rice-Nakagami random variable, are provided in the Appendix.

Statistical analysis for measuring sensor performance

When features are described probabilistically, it is subsequently possible to derive statistical metrics of sensor performance. There are actually four probabilities that are generally of interest:

1. P_{cd} = probability that noise only is present when the sensor determines that noise only is present (a correct dismissal)

2. P_{fa} = probability that noise only is present when the sensor determines that a target is present (a false alarm)

3. P_{fd} = probability that a target is present when the sensor determines that noise only is present (a false dismissal)

4. P_d = probability that a target is present when the sensor determines that a target is present (a correct detection)

These probabilities are given by the following integrals:

$$P_{\mathrm{cd}} = \int_{0}^{\beta} f_0(x)dx = 1 - P_{\mathrm{fa}}$$

$$P_{\mathrm{fa}} = \int_{\beta}^{\infty} f_0(x)dx = 1 - P_{\mathrm{cd}}$$

$$P_{\mathrm{fd}} = \int_{0}^{\beta} f_1(x)dx = 1 - P_{\mathrm{d}}$$

$$P_{\mathrm{d}} = \int_{\beta}^{\infty} f_1(x)dx = 1 - P_{\mathrm{fd}}$$

where: f_0 = pdf of the noise signal alone,

f_1 = pdf of the target and noise signals together, and

β = the detection threshold set by the detection algorithm.

A detection algorithm computes an appropriate β that ideally optimizes sensor performance by increasing probability of detection and decreasing probability of false alarm. Some examples (which are currently implemented in EASEE) include:

1. Neyman-Pearson (constant false-alarm rate) criterion
2. absolute threshold detection
3. relative threshold detection
4. error minimization
5. Bayes risk minimization

As the integrals above suggest, various probabilities of interest are computed by manipulating pdfs of signal and noise. In fact, integrations of pdfs below and above a threshold value are the cumulative distribution function (cdf) and the complementary cumulative distribution function (ccdf), respectively. Thus, we may also write:

$$P_{\mathrm{cd}} = \int_{0}^{\beta} f_0(x)dx = F_0(\beta)$$

$$P_{\mathrm{fa}} = \int_{\beta}^{\infty} f_0(x)dx = F_0^{\mathrm{C}}(\beta) = 1 - F_0(\beta)$$

$$P_{\mathrm{fd}} = \int_{0}^{\beta} f_1(x)dx = F_1(\beta)$$

$$P_{\mathrm{d}} = \int_{\beta}^{\infty} f_1(x)dx = F_1^{\mathrm{C}}(\beta) = 1 - F_1(\beta)$$

where: f = pdf,

F = cdf, and

F^{C} = ccdf

with the subscripts 0 and 1 denoting the pdf, cdf, or ccdf of the noise signal and combined target and noise signal, respectively. Strictly speaking, the cdf is actually the integral of the pdf from negative infinity to the threshold value; however, the integral from zero would only be negligibly different (if at all) when signals are represented with appropriate statistical models, whose measure limits to zero (or is even nonexistent) in the negative domain, since negative signal power or concentration is impossible. The aforementioned detection algorithms also compute the quantile (or inverse-cdf) function when determining signal-power (or concentration) thresholds corresponding to specified probabilities.

To obtain a particular statistical representation, both the type of random variable and the values for its parameters are needed. While the pdf and cdf for each type of random variable are coded to use its unique parameters (e.g., location, shape, scale, etc.), it is useful to also describe random variables with the universal parameters, mean and variance. For two-parameter random variables, closed-form expressions may be found to convert its unique parameters to/from mean and variance.

Finally, the signal pdfs within the integrals above are often joint pdfs, since multiple signals of interest and noise may arrive at a sensor simultaneously. When multiple signal sources are present within a sensor's vicinity, random variables representing each signal source must be summed at the receiver. Depending on the type of random variables being summed, computing the exact summation can be complicated and intensive. Thus, obtaining a quick approximation may be more practical. A couple of approximating approaches and the most efficient method for computing exact results are described later in this report.

3 Software Implementation

The EASEE software design is formulated within the conceptual framework of object-oriented programming in the Java language. For a comprehensive description on the object-oriented structure and approach to signal transmission and sensing in EASEE, it is recommended to consult Wilson et al. (2009). Here, only a more thorough and updated explanation of the *signal model objects* will be given.

Signal model objects

As previously described, random environmental effects on transmitted and received features are accounted for by representing them as random variables. Parametric descriptions of these random variables are programmed in EASEE within Java classes. These Java classes are denoted as *signal models* and instances of them are *signal model objects*. Signal model objects are key in the architecture of EASEE, since they are what are actually *transmitted*, *received*, and *processed*, as outlined in Figure 1. The purple arrows in the figure specify where signal models are passed from one of the five components (illustrated in boxes) to another. Specifically, the *feature generator* produces a signal model object that is inputted and then altered by the *feature propagator* and *feature sensor* before it is finally analyzed by the *feature processor* to produce an inference, which represents desired information derived from the data features such as probability of detection or error of target location estimates.

Signal model classes contain methods for the various, previously described statistical operations necessary for making probabilistic predictions on sensor performance, including:

1. setting the mean and variance
2. converting from mean and variance to unique parameter values
3. computing the pdf, cdf, and quantile
4. summing a random variable described by an instance of the class with another one.

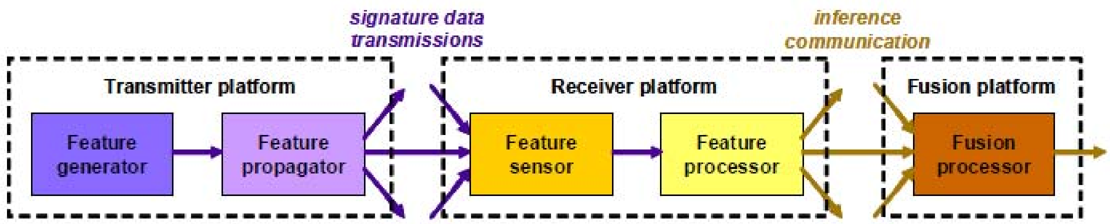


Figure 1. Generalized flow diagram for transmitting, receiving, and processing signature data features in EASEE.

These signal model objects also store arrays of parameter values defining different random variables of the same type, which may later be distributed across a terrain grid. Computations for each array component are parallelized when possible to improve computational efficiency. As mentioned earlier, EASEE currently has implementations for five continuous statistical models—namely, the Gaussian, lognormal, exponential, gamma and the $X \rightarrow X^2$ transformed Rice-Nakagami—as well as a discrete model. Each of these models is represented via its own signal model class.

Signal model inheritance relationships

The overall signal-transmission and sensing process in EASEE is described generally enough to apply for all signal modalities described by potentially very different statistics so that each of the generalized components in the above figure is coded to work with an abstract representation of all statistical models. This abstract representation is the parent abstract Java class of all signal models and is called the *abstract signal model*. It defines abstract methods for the various required statistical operations, which are then individually implemented or overridden by subclass signal models representing different statistical models. Figure 2 shows the inheritance relationships of the signal models currently available in EASEE.

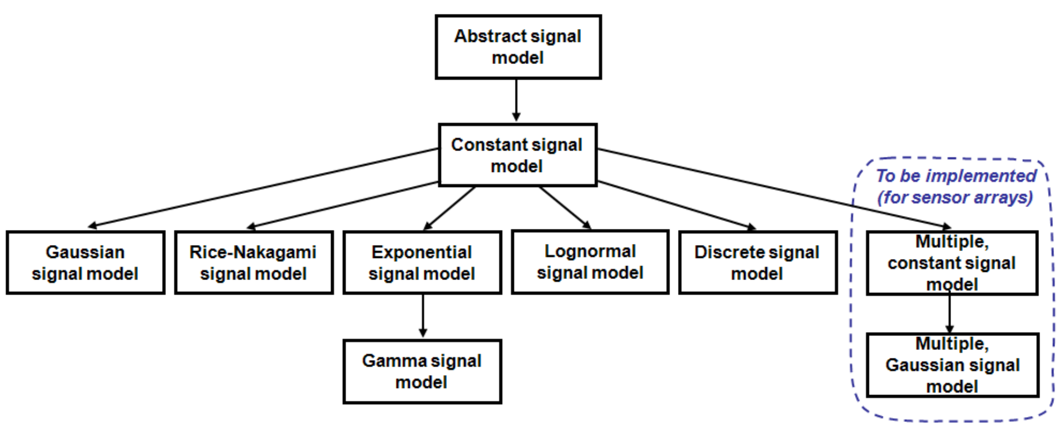


Figure 2. Inheritance tree of statistical signal models.

The *constant signal model* that extends the base abstract signal model represent constant (i.e., time-invariant) signals involving only one variable for the signal mean. A collection of variable signal models extend the constant signal. Presently, the inheritance relationships are organized so that deeper subclasses add more characteristics to their parent classes. Subclasses “inherit” and then modify the fields and/or methods of its parent class. For example, nonconstant signal models allow nonzero variance; whereas, the constant signal model does not. The gamma distribution generalizes the exponential distribution, which is, in fact, a special case of the gamma distribution with shape parameter equal to 1 and the scale parameter equal to the mean.

It may be useful, however, to structure inheritance relationships differently. For example, it may be useful to somehow categorize various distributions into similar statistical families so the sum of multiple closely related random variables may be approximated by a parametric pdf within the same family. One of the simplest methods for summing multiple random variables (described later) may, in fact, be made more efficient if inheritance is organized in such a way. Otherwise, it may be more appropriate to use interfaces to delineate statistical families within the current signal model inheritance tree, which can be used to make the described approximation method for summing random variables more efficient.

While the generalized components in the simulation scheme are *coded* to accept and return abstract signal models, specific subclass signal models will actually be used in actual simulations as appropriate via polymorphism. This programming approach using inheritance and polymorphism greatly improves software efficiency and the potential for EASEE to continuously develop and include new capabilities. Namely, existing signal-generation, propagation, and processing algorithms would automatically operate on all new signal models that extend the abstract signal model and, conversely, new processing methods that work with the abstract signal model would instantly apply to all existing and future subclass signal models.

Representing a specific feature via a signal model

While the signal models fully describe a parametric statistical model, they are not explicitly associated with any particular signature data feature. Features are separately defined as Java-enumerated types, whose representation by a signal model are assigned by a given subclass implementa-

tion of the feature generator as appropriate. Thus, the “signal data transmissions” passed from one generalized component to another in Figure 1 are essentially “packets” of information assembled by the feature generator that lists transmitted features and their corresponding signal-model representations. For organization and utility, features are defined and grouped in different Java enum classes, typically by modality. For instance, all seismic features and all radiofrequency features are enumerated in their own, separate classes. When appropriate, further distinctions are defined within a single signal modality such as with acoustic features, which have separate enum classes for acoustic octave bands (both standard and third-octave) and linearly spaced spectral bands.

4 Sums of Random Variables

When simulating multiple signals of interest and noise arriving at a sensor, probability-of-detection calculations are performed on the pdf of the sum of multiple signals. Thus, the final pdf that is analyzed at the sensor represents a sum of multiple random variables, where each individual signal is described by a unique pdf. The calculation is trivial for summing multiple Gaussian random variables, since the pdf of the sum is another Gaussian pdf with mean and variance equal to the sum of the means and variances, respectively, of each contributing random variable. In fact, the mean and variance of the sum of any (even potentially dissimilar) types of random variables is always, by definition, the sum of the means and variances of each summed random variable. (This property is referred to in the rest of this report by saying that the means and variances of the summed random variables are "conserved.") Yet, despite this property, implementation for summing multiple non-Gaussian random variables is not straightforward because an analytical form for the pdf of the sum is usually not available.

A very simple scheme

One of the simplest methods for representing the sum of multiple random variables of related types is to approximate the sum with a pdf that conserves the mean and variance of the original contributions. The pdf for the sum may be the same as one of the original random variables, or a generalized form of the pdf of the original variables. This approach is exact if all the summed random variables are constant or Gaussian, and also for some other very particular cases, such as for gamma random variables with the same scale parameter. Otherwise, it is only an approximation that is not necessarily accurate in all cases. For instance, Stewart et al. (2007) observe that this approach is only reasonably accurate when summing two gamma random variables if the shape parameters are not lower than 0.1 and the scale parameters do not differ by more than a factor of 10. Accuracy of approximating the sum of two lognormal random variables by another lognormal random variable may be adequate in one tail of the summed distribution but not the necessarily the other (Mehta et al. 2007). The accuracy of this approximation with other distributions such as the transformed Rice-Nakagami is unverified. While the method seems to make intuitive sense, the accuracy of the approach is difficult to confirm or predict

for various distributions and cases. It is reasonable to assume that there are limitations and that it is not good in all cases.

However, this method is a very simple alternative to other more robust yet computationally intensive approaches and much easier to implement. It is also important to note that parametric pdfs representing various signals are themselves only estimates to begin with because both the models and the parameters in the models used to generate them are approximate due to idealizations of the signal physics and uncertainties in the weather, ground conditions, source signature and directivity, noise background, etc. Especially if it will greatly increase computational intensity, it is not necessarily worthwhile to compute the sum of multiple, imprecise random variables with very high precision.

Although the exact sum of multiple random variables of related type cannot typically be represented by an analytical pdf, this method essentially suggests selecting one that will serve as a good approximation with mean and variance conserved. Generally, the sum is best approximated by the pdf of one of the summed random variables, but sometimes it is better to use a pdf that is different but still a generalized form of the summed random variables. For example, the sum of multiple exponential random variables would be better approximated by a gamma distribution rather than another exponential distribution, where means and variances are conserved. In fact, this would be exact if the exponential random variables were all identical (with the same mean). Likewise, the summation of multiple random variables of different but related type is represented as a pdf of the most generalized form of the related random variables. An example of this is the sum of an exponential random variable and a gamma random variable, which would be a gamma random variable with the means and variances conserved. Here, the exponential and gamma distributions are closely related, belonging to the exponential class of density functions, where the gamma distribution with shape parameter equal to 1 reduces to the exponential distribution with a mean equal to the scale parameter (i.e., the exponential distribution is a special case of the gamma distribution). So, in fact, the sum of exponential and gamma random variables is essentially just a sum of gamma random variables.

Conceivably (but not necessarily), the method could well approximate the sum of many different kinds of random variables, so long as a parametric distribution that generalizes a diverse set of random variables is available.

For instance, the sum of any combination of two or more of the lognormal, exponential, Weibull, and gamma distributions may be well approximated by a generalized gamma distribution with means and variances conserved, since the aforementioned, well-known distributions are actually special cases of the generalized, three-parameter gamma distribution. Indeed, there are other three-parameter distributions that generalize various well-known distributions such as the generalized and skew normal distributions, both of which can mimic other skewed distributions like the gamma, lognormal, and Weibull distributions but also includes the normal distribution as a special case.

Though there is an added complication in representing the sum of random variables with a three-parameter distribution, since the three parameters must be estimated numerically via maximum likelihood or method of moments, it is possible and potentially worthwhile if the process proves to be relatively less computationally intensive (Stacy and Mihram 1965; Ashkar et al. 1988; Varanasi and Aazhang 1989). When using the method of moments, it is helpful to notice that the third unstandardized central moment can be computed exactly for the sum, since they are also "conserved" like the means and variances, which, in fact, are the first raw moment and the second unstandardized central moment, respectively. This results in three moments that can be computed exactly for the sum, which can be used to estimate the required three parameters to obtain the generalized distribution. Otherwise, it may be advantages to compute a few particular cumulants of the generalized distribution of the sum to estimate its parameters instead, since all cumulants are conserved.

As described, this method only applies for summing multiple random variables of *related type*. If no generalized distribution for two or more diverse random variables is available, it is impossible to predict the appropriate analytical pdf that best approximates their sum. For example, the lognormal and the transformed Rice-Nakagami random variables are not of related type and do not have a parametric distribution that generalizes both of them, so this method does not apply in this case.

Finally, this method is both commutative and associative in that the order in which multiple random variables are summed together does not affect the final representation of their sum. This follows first from the fact that the summation of means and variances (and the third unstandardized central moment, if necessary) is itself commutative and associative. Then, the

method selects the analytical pdf of the same type or of a more generalized type than that which describes the two summed random variables, regardless of which one is presented first in the method's algorithm.

Using the Laplace approximation

The Laplace approximation is another approach that introduces the approximation at a different stage in the summation of random variables. Rather than approximating the resulting sum of multiple random variables with an appropriate analytical pdf as in the previously described method, it is possible to approximate each non-Gaussian random variable addend by a Gaussian pdf. Then, the sum of these Gaussian approximations may be computed exactly, since the exact sum of multiple Gaussian random variables is simply another Gaussian random variable with means and variances conserved.

Since many familiar, unimodal random variables may be roughly approximated by a Gaussian pdf, the Laplace approximation is widely used in many situations requiring manipulation of multiple random variables that must be or is greatly preferred to be Gaussian. In our case, obtaining a good Gaussian approximation of multiple random variables greatly simplifies their summation by avoiding the use of a more computationally intensive, numerical approach. It is also likely that the Laplace approximation is within the precision errors inherent in the idealizations and uncertainties of the models and parameters used in simulating the signal-transmission and sensing process.

Since the Laplace method involves the second-degree Taylor expansion of the given, non-Gaussian pdf's logarithm centered on the mode, both the mode and the second derivative of the pdf's logarithm at the mode must be computed. Often these can be calculated with simple, closed-form expressions. If not, quick numerical computations can be devised noting that the mode is defined as where the maximum of the pdf occurs. Details on the theory and definition of the Laplace approximation can be found in Bishop (2006).

The convolution approach

Though it is the most computationally intense, it is possible to always compute the pdf of the sum of two independent random variables via the convolution of their pdfs. Since all pdfs are compactly supported and

locally integrable, the convolution of two pdfs, f and g, denoted as $f * g$, is the following well-defined and continuous integral transform:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{\Re} f(\tau) g(t-\tau) d\tau$$

The preferred method for computing the convolution is via the forward and inverse discrete Fourier transforms, for which fast Fourier transform algorithms are available. According to the convolution theorem, the Fourier transform of a convolution is the pointwise product of the Fourier transforms of the two functions being convoluted together:

$$F[f * g] = k \cdot F[f] \cdot F[g]$$

where k is a constant that depends on the normalization of the Fourier transform and $F[f]$ denotes the Fourier transform of f. Then, the inverse Fourier transform may be applied to obtain the convolution, $f * g$:

$$f * g = F^{-1}[F[f * g]] = F^{-1}[k \cdot F[f] \cdot F[g]] = kF^{-1}[F[f] \cdot F[g]]$$

Since the characteristic function of a random variable is a Fourier transform of its pdf and is available for every type of random variable, it is then possible to compute the sum of multiple random variables using the inverse Fourier transform and characteristic functions by the general formula above.

By using the convolution theorem and the fast Fourier transform (FFT) algorithms, an implementation for computing the sum of multiple random variables via the numerical convolution of two pdfs may be made practical. In practice, the FFT algorithm must be implemented carefully to avoid errors from discretization and truncation of integrals as much as possible. This can be quite difficult, but, if done correctly, the convolution approach would make it possible to sum multiple random variables of even vastly different type. It is also commutative and associative, as required.

Central limit theorem

As more independent random variables are summed, the joint random variable monotonically converges into a Gaussian distribution given certain conditions. By the classical central limit theorem, the sum must be of many independent and identically distributed random variables, each having positive variance with both mean and variance being finite.

However, the theorem actually also holds for non-identically distributed random variables if Lyapunov's condition is satisfied, which further implies that Lindeberg's condition is also met (Ash and Doléans-Dade 2000). Though the rate of convergence depends on the type(s) of the identical or non-identically distributed random variable addends, it is conceivable that the sum of even three to five non-Gaussian random variables may already be well approximated by a Gaussian distribution. Hence, in cases where enough random variables are added together, where at least one of them is non-Gaussian, the sum may be simply represented by a Gaussian distribution with means and variances conserved. Any implementation for summing multiple random variables should take this limit theorem into account to improve both accuracy as well as computational speed. When the theorem applies, the sum should be directly represented as Gaussian with means and variances conserved rather than computing many Laplace approximations, numerically evaluating many convolutions, or using a different pdf for the final representation.

5 Numerical Issues

Although most statistical operations required for the pdfs described in this report are relatively straightforward to program, it is helpful to illustrate some of the operations that make use of certain numerical algorithms. It is also worthwhile to describe how more complicated operations must be programmed carefully so that they compute properly using floating-point arithmetic on a computer. For the latter concern, only a selection of examples is presented to illustrate some common problems that may arise. The Appendix includes further details.

Computing the quantile using the bisection method

With the exception of the exponential model, for which a closed-form formula for evaluating the quantile is available, all other statistical models are coded to numerically solve for the quantile, x, by approximating the root, y_0, of the following function, $Q(y)$, using the bisection method for a given $P = \mathrm{cdf}(x)$ with the tolerance set to the lowest representable number in binary double (64-bit) floating-point precision (approximately -1.8 × 10^{308}):

$$Q(y) = \mathrm{cdf}(y) - P$$

where the root, y_0, is equal to the quantile, x:

$$Q(y_0) = \mathrm{cdf}(y_0) - P = \mathrm{cdf}(y_0) - \mathrm{cdf}(x) = 0 \Leftrightarrow \mathrm{cdf}(y_0) = \mathrm{cdf}(x) \Leftrightarrow y_0 = x$$

The lower- and upper-bounds used for the bisection method is set to span the entire domain supported by a particular statistical distribution that is representable in binary double (64-bit) floating-point precision. Specifically, the lower- and upper-bounds for the lognormal, gamma, and $X \to X^2$ transformed Rice-Nakagami models are set to zero and the highest representable number (approximately 1.8 × 10^{308}). For random variables with real-number support like the Gaussian, the lower- and upper-bound is set to the lowest and highest representable number (approximately -1.8 × 10^{308} and +1.8 × 10^{308}), respectively. The bisection method used to solve for the quantile requires that the cdf be always computable within the bounds. Since the bounds are set to span very large domains, the cdf must be programmed carefully so that it remains computable for all input values within the domain. Concerns related to how the cdf should be programmed to satisfy this required condition are discussed in the last two

sections of this chapter ("Numerical stability and well-posed problems" and "Illustrative examples of numerically stable statistical computations").

Although the bisection method only converges linearly (i.e., the absolute error of the true and estimated roots is halved at each step), it is preferred here because of it robustness. Unfortunately, other methods with higher convergence rates such as the Newton and Secant methods fail due to the nature of the cdf curve, which is asymptotically a horizontal line (with the first derivative equal to zero) at both ends. An optimized algorithm with a mixture of different methods is conceivable but does not necessarily improve the convergence rate appreciably.

Computing the cdf using Gauss-Legendre quadrature on the pdf

Usually there is an expression for the cdf that could be directly implemented, but, if it is complicated (e.g., it contains a special function that is difficult to program), numerical integration of the pdf may be a simpler alternative, although it is generally more computationally intensive:

$$\mathrm{cdf}(x) = \int_0^x \mathrm{pdf}(t)\,dt$$

The Gauss-Legendre quadrature rule is a very efficient method for performing the integration. Basically, the method approximates the definite integral of a function via a weighted sum of function values at specified points within the domain of integration. More details about stably implementing this method for arbitrary integration domains are provided in the Appendix.

In addition to the lower- and upper-bounds of the integration, the method requires specification of the number of weighted function values, N, to be summed. The appropriate number of points for the Gauss-Legendre rule, N, so that the integration of the pdf is computed to a sufficient level of precision depends on the nature of the pdf curve over the specified range of integration. Since the theory behind the N-point Gauss-Legendre rule ensures that it computes *exact* integrals for the class of polynomials of degree $2N$-1 or less, one must essentially predict the degree of the polynomial that adequately approximates the pdf over the specified range of integration to find the appropriate N. As N is increased, the numerical integration becomes more computationally intensive (and time-consuming), so it is

important to choose an N that is not greater than what suffices to obtain results of sufficient precision.

To minimize the number of points necessary in the Gauss-Legendre rule and still compute $\mathrm{cdf}(x)=\int_0^x \mathrm{pdf}(t)\,dt$ with sufficient precision, it is helpful to minimize the range of integration as much as possible by programming cdf computation differently depending on x:

$$\mathrm{cdf}(x)=\begin{cases} 0, \text{ for } \mathrm{pdf}(x)=0 \text{ and } x<\text{mean} \\ \int_l^x \mathrm{pdf}(t)\,dt, \text{ for } \mathrm{pdf}(x)>0 \text{ and } x<\text{mean} \\ 1-\int_x^u \mathrm{pdf}(t)\,dt, \text{ for } \mathrm{pdf}(x)>0 \text{ and } x\geq\text{mean} \\ 1, \text{ for } \mathrm{pdf}(x)=0 \text{ and } x>\text{mean} \end{cases}$$

where l and u are approximations of the lowest and highest values for x with nonzero pdf values, respectively. More precisely, l and u are computed as the first integer standard deviate below and above the mean with a pdf value of zero, respectively. If l and/or u is beyond the given statistical distribution's representable support domain, it is set as the lowest and highest representable value of the support domain. Then, empirical investigations suggest that the 50-point Gauss-Legendre rule would be sufficiently precise for computing any desired cdf value (Figure 3). Although all parameters cannot be tested, it is reasonable to assume that the general behavior of all pdfs of the same type is similar enough that the 50-point rule would always suffice.

This approach is currently used for the $X \to X^2$ transformed Rice-Nakagami model to avoid implementing the first-order Marcum Q-Function in its cdf expression. It is also included in the gamma model as an alternative in the potential event that the implementation for the direct approximation fails for certain inputs. As it turns out, some care must be taken in programming floating-point implementations of certain pdfs and cdfs (including the gamma cdf), as will be discussed in the following sections.

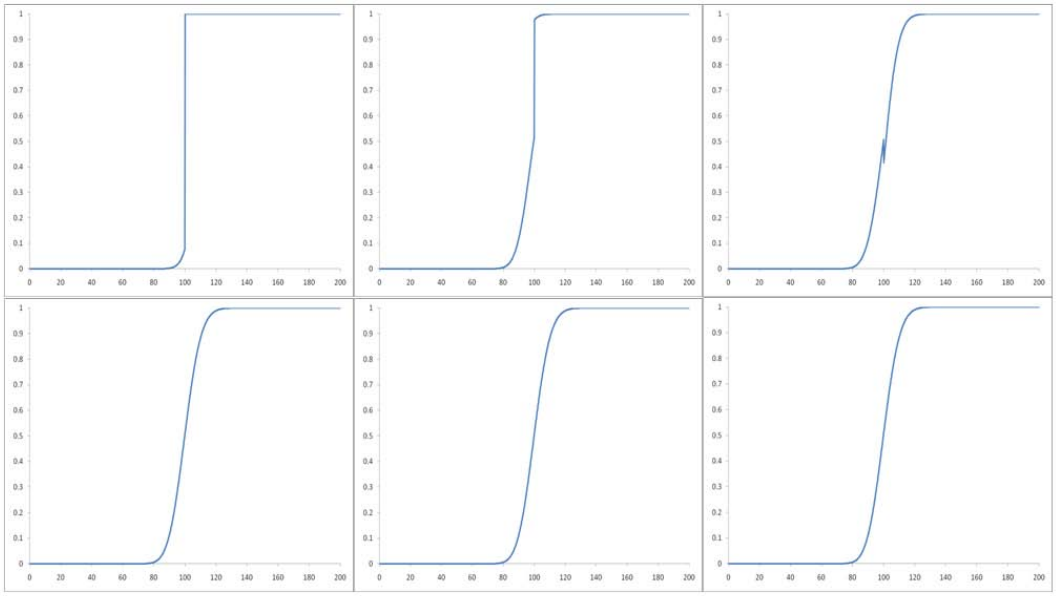


Figure 3. Comparison of cdf generated using different *N*-point Gauss-Legendre rules to integrate the $X \rightarrow X^2$ transformed Rice-Nakagami pdf with mean = 100 and var = 70.
Vertical axis: pdf(x); horizontal axis: x. From left to right on the top row: N = 2, 5, and 10. From left to right on the bottom row: N = 20, 50, and 100. As N is increased, accuracy of the pdf integration is improved and a smooth cdf curve is generated.

Numerical stability and well-posed problems

When programming the various statistical operations, it is important to code calculations so that they function properly when performed using floating-point arithmetic on a computer. For programming very simple calculations, it is not necessarily important to understand the mechanics of floating-point arithmetic. It is, however, essential to understand when formulating more complicated functions and numerical algorithms. When a calculation is programmed to function properly within the limitations of floating point, it is called *numerically stable*. Goldberg (1991) has a much more comprehensive treatment of the subject than that which is provided here.

The basic problem is caused by computers with limited memory that can only represent (and, therefore, work with) a finite, discrete set of numbers. Floating point describes a system for allocating computer memory efficiently so that a very large set of numbers can be made "representable." A more thorough background on floating point is provided in the Appendix.

A variety of issues due to floating point are well known. First, there is round-off error, which is the difference between the calculated approximation of a number and its exact mathematical value. This type of error is in-

troduced when the inputs and outputs of an arithmetic operation must be rounded to the nearest representable number. Typically, round-off errors are insignificant in simple computations, except in relatively rare instances (e.g., when two large, nearly equal numbers are subtracted, causing the number of accurate significant digits in the difference to be greatly reduced, sometimes called a *catastrophic cancellation*). But, small round-off errors may accumulate unacceptably when performing operations that require a sequence of many calculations (e.g., large summations, matrix inversion, eigenvector computation, and solving differential equations).

Another concern is more critical in our case, particularly when programming the pdf, cdf, and quantile. Given that the final result of a computation is indeed representable, it is important that the results of the various arithmetic operations within that computation remain within the given floating-point representation's domain of representable numbers. Ideally, a computation should be programmed so that this condition is met for *all* representable inputs. When this condition is not met, the final result is typically either very inaccurate or not computable at all, since an arithmetic operation has resulted in an overflow or underflow (which is denoted by saying that the computation has suffered from *premature overflow* or *underflow*, respectively). Then, the computation could unpredictably return an undefined or infinite number (e.g., $+\infty$, $-\infty$, or NaN), although the true, mathematical solution to the computation is within the representable range. For example, computation of the average of two nonzero numbers, x and y, that are within the representable domain using the formula, $(x + y) / 2$, may return $+\infty$ or $-\infty$ if $(x + y)$ overflows, although the average would always lie within the representable range.

To address this latter issue, it is helpful to notice that sometimes a single calculation can be achieved in several ways. In the previous example for computing the average of the two numbers, x and y, it is better to program the distributed formula, $x / 2 + y / 2$, which does not have the overflow problem. But, this formula is now susceptible to underflow by the terms, $x / 2$ or $y / 2$, for small enough x or y. Since premature underflow would only present, at most, an absolute error of twice the smallest representable number, it is more important to program against premature overflow than premature underflow *in this case*—for other situations when underflow may present a division-by-zero operation, premature underflow is much more important to prevent. The final, numerically stable algorithm for averaging the two numbers, x and y, would compute the distributed

formula, $x / 2 + y / 2$, *only* if the original formula, $(x + y) / 2$, results in an overflow.

If a computation is rather complicated, a careful, numerically stable floating-point computer implementation of it can be quite difficult to program. Although various algebraic manipulations are commonly used, each computation is unique and requires different techniques. While it is easier to program a computation that must be stable only within a limited domain of inputs, the inputs that the computation must be able to handle may be difficult to know or predict.

Illustrative examples of numerically stable statistical computations

Since they share some general similarities, many statistical formulae suffer from common issues. For instance, many pdfs are of the following form:

$$\mathrm{pdf}(x) = ab\exp(-cd) = \frac{ab}{\exp(cd)}.$$

A simple, direct implementation of the pdf expression may not normally present issues with ordinary values for a, b, c, and d. However, this formulation is certainly not robust for *all* representable a, b, c, and d. Namely, the numerator and denominator may both overflow, although their ratio is a perfectly ordinary value (that always lies within the representable range).

This potential problem is easily remedied by a single, straightforward algebraic rearrangement. Specifically, the division of two potentially large numbers, A and B, is better encoded via the difference of their logarithms:

$$\frac{A}{B} = \exp[\ln(A) - \ln(B)]$$

so that the ratio becomes computable for a much more expansive domain of A and B, since it is only required that their *logarithms* not overflow. Using this rearrangement, the general pdf expression above becomes:

$$\mathrm{pdf}(x) = \frac{ab}{\exp(cd)} = \exp\left[\ln\left(\frac{ab}{\exp(cd)}\right)\right] = \exp[\ln(ab) - \ln(\exp(cd))] = \exp[\ln(ab) - cd] = \exp[\ln(a) + \ln(b) - cd]$$

It is important to note here that not all overflows are necessarily an issue but that they become problematic only when inaccuracies may result. For example, the potential overflow of the term, cd, in the above rearrangement is not an issue because it would always imply that the pdf is zero

and, in fact, actually results in a zero to be returned for the pdf. (It is helpful to notice here that the rearranged pdf would always equal zero when the term, *cd*, becomes large enough even *before* it overflows.) Overflow of *A* and/or *B* in the previous expression is troublesome because it does not imply anything about the true value of *A / B* and inaccurately returns 0, +∞, or NaN.

Since computing exponentials and logarithms are more computationally expensive, it is better to calculate the stable rearrangement as an alternative only when the floating-point computation of *A / B* fails, *if* the simpler formulation is only rarely unstable. In certain cases, however, it may be that the straightforward formula, *A / B*, is actually more often unstable than it is stable. This is true, for example, with the gamma cdf, which will be described shortly. In these cases, it would be more efficient to just compute the stable rearrangement directly. The Appendix includes specific details on how to apply this rearrangement for each type of distribution.

As just stated, this technique is useful in coding the gamma cdf, defined for $x \geq 0$, $\alpha > 0$, and $\beta > 0$:

$$\mathrm{cdf}(x) = P\left(\alpha, \frac{x}{\beta}\right) = \frac{\gamma\left(\alpha, \frac{x}{\beta}\right)}{\Gamma(\alpha)}$$

where *P* is the regularized incomplete gamma function:

$$P(s,x) = \frac{\gamma(s,x)}{\Gamma(s)} = \frac{1}{\Gamma(s)} \int_0^x r^{s-1} e^{-r}\, dr$$

and Γ and γ are the gamma function and the lower incomplete gamma function, respectively.

In this formula, the individual terms, $\gamma\left(\alpha, \frac{x}{\beta}\right)$ and $\Gamma(\alpha)$, overflow at modest values of x, α, and β, although the regularized incomplete gamma function itself has the limiting values:

$$P(s,0) = 0 \text{ and } P(s,\infty) = 1.$$

Thus, it is also helpful in this case to implement the exponentiation of the difference of the logarithms of the two terms:

$$\mathrm{cdf}(x) = P\left(\alpha, \frac{x}{\beta}\right) = \frac{\gamma\left(\alpha, \frac{x}{\beta}\right)}{\Gamma(\alpha)} = \exp\left[\ln\left(\frac{\gamma\left(\alpha, \frac{x}{\beta}\right)}{\Gamma(\alpha)}\right)\right] = \exp\left[\ln\left(\gamma\left(\alpha, \frac{x}{\beta}\right)\right) - \ln(\Gamma(\alpha))\right].$$

Though they are used less frequently, various other manipulations are useful in more specific circumstances (and often in combination with the first technique). For instance, after using this first technique, the Gaussian cdf still contains a term that resembles the following:

$$\ln(a+b).$$

For all representable a and b, the true value of this term is always within the representable range, but $+\infty$ would be returned whenever $(a + b)$ overflows. This issue can be avoided by programming the following:

$$\ln(a+b) = \ln\left[\left(\frac{a+b}{c}\right)c\right] = \ln\left[\left(\frac{a}{c} + \frac{b}{c}\right)c\right] = \ln\left(\frac{a}{c} + \frac{b}{c}\right) + \ln(c)$$

where c is set as the largest representable number in the given floating-point system.

A similar distributive approach is used in multiple other situations, including the formula for the gamma pdf, which presents the following general expression:

$$ab - c.$$

When implemented directly, the subtraction by c is ignored whenever the term, ab, overflows and results in the entire expression being automatically equated to $+\infty$ or $-\infty$. This is unfortunate if c is of the same sign as ab, since the entire expression may very well be within the representable range (if c is of large enough magnitude). This problem can be easily avoided by programming the following instead:

$$ab - c = d\left(\frac{ab-c}{d}\right) = d\left(\frac{ab}{d} - \frac{c}{d}\right)$$

where d is selected to be large enough so that the term, $\frac{ab}{d}$, does not overflow.

In many instances, it is only important to simply pay attention to the order of operations. This, in fact, applies for programming the term, $\frac{ab}{d}$, in the

technique explained above. It also applies in many other situations, including when programming a term in the lognormal cdf, which is of the following form:

$$\frac{a}{bc}.$$

For even a simple expression like this, there are many ways to program it:

$$\frac{a}{bc}=\frac{a}{(bc)}=\left(\frac{a}{b}\right)\left(\frac{1}{c}\right)=\left(\frac{1}{b}\right)\left(\frac{a}{c}\right)=a\left(\frac{1}{(bc)}\right)=\frac{\left(\frac{a}{b}\right)}{c}=\frac{\left(\frac{a}{c}\right)}{b}.$$

With some quick investigation, it is relatively easy to confirm that (usually) only *one* of these implementations (if any) is stable given certain limitations on the domain of *a*, *b*, and *c*. But, in a couple of special cases of this general expression, it is *always* best to implement them in the following way:

$$\frac{a}{b^2}=\frac{\left(\frac{a}{b}\right)}{b}$$

and

$$\frac{a^2}{b}=a\left(\frac{a}{b}\right).$$

Simply changing the order of operations is often sufficient throughout parts of various statistical formulae; however, attention to this detail is quite easily overlooked. Application of this technique in specific cases is described within the Appendix.

When no particular ordering of operations is universally stable for the given domains of *a*, *b*, and *c*, it is always possible to resort to the following general rearrangement that is stable for *all* representable *a*, *b*, and *c*:

$$\frac{a}{bc}=\exp\left[\ln\left(\frac{a}{bc}\right)\right]=\exp[\ln(a)-\ln(b)-\ln(c)].$$

Finally, it is helpful to note certain consequences of finite-precision arithmetic when analyzing and implementing stable mathematical expressions in floating point. Essentially, any floating-point system represents a finite, discrete set of numbers along the real number line that retains the same *relative* difference only between each representable number (Figure 4).

This results in additions or subtractions by sufficiently small numbers to be ignored because the floating-point system is not precise enough to represent the difference. This observation is useful when programming, for instance, the following general expression, which arises in lognormal parameter conversions:

$$\ln(a + bc)$$

which becomes the following for sufficiently large differences in magnitude between *a* and *bc*:

$$\ln(a + bc) \xrightarrow{|a|<<|bc|} \ln(bc) = \ln(b) + \ln(c).$$

This is a simpler alternative to the earlier, distributive technique:

$$\ln(a + b) = \ln\left[\left(\frac{a+b}{c}\right)c\right] = \ln\left[\left(\frac{a}{c} + \frac{b}{c}\right)c\right] = \ln\left(\frac{a}{c} + \frac{b}{c}\right) + \ln(c)$$

if it can be known that there will always be a sufficiently large magnitude difference between the added or subtracted terms.

This observation is often useful in analyzing the stability of an implementation and concluding that certain potential underflow issues are acceptable to ignore. Specifically, the magnitude difference of a given representable number and an underflowing number may be large enough that the underflow to zero would not affect their summation, since the contribution by the underflowing number would be ignored anyways even if it *was not* necessarily underflowing. However, there are situations when it is prudent

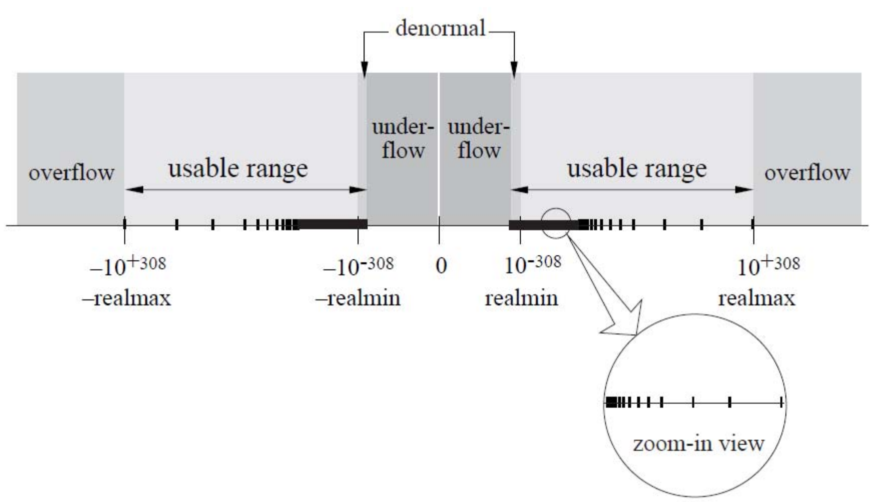


Figure 4. Floating-point number line (Recktenwald 2006).

to check and fix a complicated implementation for potential underflow issues, particularly if the underflow would result in a division or multiplication by zero. For example, it would be better to program the following two expressions as shown:

$$(ab)(cd) = \exp[\ln((ab)(cd))] = \exp[\ln(ab) + \ln(cd)] = \exp[\ln(a) + \ln(b) + \ln(c) + \ln(d)]$$

and

$$\frac{(ab)}{(cd)} = \exp\left[\ln\left(\frac{(ab)}{(cd)}\right)\right] = \exp[\ln(ab) - \ln(cd)] = \exp[\ln(a) + \ln(b) - \ln(c) - \ln(d)]$$

so that underflow by the terms, *ab* and/or *cd*, would not result in potential inaccuracies due to the expressions being automatically computed as 0, +∞, -∞, or NaN. The Appendix includes more detailed explanations of potential underflow issues for particular statistical computations.

There are obviously many more algebraic manipulations that could be useful in different situations; the listed techniques here are by no means exhaustive. They are, however, what is most helpful for carefully programming various statistical operations in floating point. As the mechanics of floating point are better understood, it becomes easier to discover other appropriate techniques and improve coding habits. Gradually, it becomes obvious that certain expressions should be coded *always* in a certain way such as:

$$\ln(ab) = \ln(a) + \ln(b)$$

and

$$\sqrt{ab} = \sqrt{a}\sqrt{b}$$

As it turns out, designing stable floating-point implementations is highly nontrivial. But, it is essential if the domain of inputs is expansive or unknown and/or if a particular numerical algorithm depends on it to function properly. In the case here, for example, stable implementations of the cdf is vital if the algorithm that solves for the quantile uses a root-finding method with bounds set to span a very expansive domain (e.g., the entire representable region). Only a broad, conceptual discussion is given here for brevity. The Appendix gives specific details for implementing a particular computation.

6 Conclusions

Given the randomness of most signal-generation and propagation processes, predictions of sensor performance are inherently probabilistic in nature. While it is important to minimize (e.g., using accurate parameters and models), various types and levels of uncertainty would still exist in virtually all situations when modeling a complicated process such as signal transmission and sensing. Especially when small variations in the environmental state dramatically alter outcomes, calculating and communicating predictions in a statistical manner is critical.

This report focused on representing target and noise signals as random variables to account for variations in sensor data (e.g., sound level, seismic energy, concentration of a chemical agent, etc.) that are essentially stochastic due to many unknown and potentially dynamic environmental variations affecting signal generation and propagation. By modeling signals statistically, the report discussed how to make *probabilistic* predictions on sensor performance. Both the software implementation of the statistical framework and some important numerical issues in programming statistical computations were also explained.

References

Andrews, L. C., and R. L. Phillips. 2005. *Laser beam propagation through random media*. 2nd ed. Bellingham, WA: SPIE.

Ash, R. B., and C. A. Doléans-Dade. 2000. *Probability and measure theory*. 2nd ed. pp 307–317. San Diego, CA: Academic Press.

Ashkar, F., B. Bobée, D. Leroux, and D. Morisette. 1988. The generalized method of moments applied to the generalized gamma distribution. *Stoch. Hydrol. Hydraul.* 2:161-174.

Baker, M. B., M. Eylander, and H. Harrison. 1983. The statistics of chemical trace concentrations in the steady state. *Atmos. Environ*. 18:969-975.

Bishop, C. M. 2006. *Pattern recognition and machine learning.* New York, NY: Springer-Verlag.

Burdic, W. S. 1984. *Underwater acoustic system analysis*. Englewood Cliffs, NJ: Prentice Hall.

Goldberg, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* 23:5-48.

Hieb, M. R., S. Mackay, M. W. Powers, H. Yu, M. Kleiner, and J. M. Pullen. 2007. *Geospatial challenges in a net centric environment: Actionable information technology, design, and implementation*, report 657816. *In* Proceedings of SPIE Defense and Security Symposium, Defense Transformation and Net-Centric Systems, edited by R. Suresh.

Lanczos, C. 1964. A precision approximation of the gamma function. *J. Soc. Ind. Appl. Math: Series B, Numerical Analysis* 1:86-96.

Limpert, E., W. A. Stahel, and M. Abbt. 2001. Log-normal distributions across the sciences: Keys and clues. *BioScience* 51:341-352.

Mehta, N. B., J. Wu, A. F. Molisch, and J. Zhang. 2007. Approximating a sum of random variables with a lognormal. *IEEE T. Wirel. Commun.* 6:2690-2699.

Nadarajah, S. 2008. A review of results on sums of random variables. *Acta Appl. Math* 103:131.140.

Pugh, G. R. 2004. *An analysis of the Lanczos gamma approximation*. Ph.D. dissertation, pp 114-116. Vancouver, Canada: The University of British Columbia.

Recktenwald, G. W. 2006. *Unavoidable errors in computing.* Accessed July 2010. <<http://web.cecs.pdx.edu/~gerry/nmm/course/slides/ch05Slides.pdf>>

Stacy, E. W., and G. A. Mihram. 1965. Parameter estimation for a generalized gamma distribution. *Technometrics* 7:349-358.

Stewart, T., L. Strijbosch, H. Moors, and P. Batenburg. 2007. *A simple approximation to the convolution of gamma distributions,* report 2007-70. A CentER Discussion Paper. ISSN 0924-7815.

Strohbehn, J. W. editor. 1978. *Laser beam propagation in the atmosphere*. New York, NY: Springer-Verlag.

Varanasi, M. K., and B. Aazhang. 1989. Parametric generalized Gaussian density estimation. *J. Acoust. Soc. Am.* 86:1404-1415.

Wilson, D. K., R. Bates, and K. K. Yamamoto. 2009. *Object-oriented software model for battlefield signal transmission and sensing*. ERDC/CRREL TR-09-17/ ADA522523. Hanover, NH: U.S. Army Engineer Research and Development Center.

Wilson, D. K., C. L. Pettit, S. Mackay, M. S. Lewis, and P. M. Seman. 2008. *Addressing uncertainty in signal propagation and sensor performance predictions.* ERDC/CRREL TR-08-21/ ADA4591357. Hanover, NH: U.S. Army Engineer Research and Development Center.

Wilson, D. K., B. M. Sadler, and T. Pham. 2002. *Simulation of detection and beamforming with acoustical ground sensors*, 50-61. In Proceedings of SPIE AeroSense Symposium, Unattended Ground Sensor Technologies and Applications IV, edited by E. M. Carapezza.

Appendix: Floating-Point Implementations of Statistical Formulae

This appendix includes specific details on floating-point implementations of the various statistical formulae described in the report. Mathematical expressions are programmed so that they always return a representable number whenever the true solution is representable for all representable parameter inputs. When this condition is met, it is said that the implementation is *numerically stable*. The representable numbers are often defined by the binary double (64-bit) floating-point precision system, which has approximately 16 decimal digits of precision and spans numbers from approximating -1.8×10^{308} to $+1.8 \times 10^{308}$. A short background on floating point is given as an introduction.

Floating point

To be able to represent the widest range of numbers possible using a fixed number of bytes in computer memory, modern machines approximate numbers to a fixed number of significant digits (called the significand), which is scaled by multiplication with a base (normally 2, 10, or 16) raised to a specified exponent:

$$\text{significant digits} \times \text{base}^{\text{exponent}}$$

This numeral system, called **floating point**, allows the radix point that separates the integer part of a number (to the left of the point) from the fractional part (to the right of the point) to be placed anywhere relative to the significant digits of the number (i.e., to "float"), making it conceptually similar to scientific notation. While several different floating-point implementations have been used in the past, the Institute of Electrical and Electronics Engineers (IEEE) has standardized the practice in IEEE 754, which is now followed by almost all modern computers.

The IEEE 754 Standard for Floating-Point Arithmetic defines several basic floating-point formats using different radixes (i.e., bases) and amounts of computer bits, of which two are most widely used in computer hardware and programming languages:

1. binary (i.e., base 2) single precision, which occupies a total 32 bits (4 bytes) and has a significand precise to 24 bits (about 7 decimal digits)
2. binary double precision, which occupies a total of 64 bits (8 bytes) and has a significand precise to 53 bits (about 16 decimal digits).

Other basic floating-point formats include binary quadruple precision (occupying 128 bits) and decimal (i.e., base 10) formats encoded using 64 or 128 bits. All basic floating-point formats also define representations of special values, including positive and negative infinities ($+\infty$ and $-\infty$), negative zero (-0) distinct from ordinary ("positive") zero, and a "not a number" value (NaN), where NaN is generated by three kinds of operations:

1. operations that produce indeterminate forms:
   a. the divisions $0/0$, ∞/∞, $\infty/-\infty$, $-\infty/\infty$, and $-\infty/-\infty$
   b. the multiplications $0\times\infty$ and $0\times\infty$
   c. the power 1^{∞}
   d. the additions $\infty+(-\infty)$, $(-\infty)+\infty$, and equivalent subtractions (i.e., $\infty-\infty$ and $-\infty-(-\infty)$)
2. real operations with complex results:
   a. the square root of a negative number
   b. the logarithm of a negative number
   c. the inverse sine or cosine of a number which is less than -1 or greater than +1
3. any operation with a NaN as at least one operand

The standard further defines how to simulate arithmetic operations (e.g., add, subtract, multiply, divide, square root, exponentiation, sine, cosine, etc.) on floating-point numbers, recognizing certain exceptional situations, which include:

1. invalid operations (e.g., square root of a negative number)
2. division by zero
3. overflow (a number that is too large to represent correctly)
4. underflow (a number that is very small and is inexact)
5. inexact (an approximated number)

Finally, the standard also defines various algorithms for rounding floating-point numbers during arithmetic and conversions, where the default method rounds to the nearest representable value or to the nearest representable even number if the number lies midway between two representable values.

Stable statistical implementations

Details given here for the Gaussian, lognormal exponential, and gamma distributions are thorough enough that a reader may understand *exactly* how they were implemented. Although the details for the $X \to X^2$ transformed Rice-Nakagami model are not as complete, the ideas and techniques used in its implementation are very similar to those of the other models. The derivation for the random variable transformation to obtain the $X \to X^2$ transformed Rice-Nakagami distribution is also provided.

When variance is set to zero, the density function does not exist and the probability distribution becomes degenerate, describing a discrete random variable whose support consists of only one value. While such a distribution is not random in the practical sense, it does still satisfy the definition of a random variable and may be described as such in order to provide a way to deal with constant values in a probabilistic framework. Namely, the probability mass function (pmf) for this case is given by:

$$\mathrm{f}(x) = \begin{cases} 1, \text{ for } x = \text{mean} \\ 0, \text{ otherwise} \end{cases}$$

Then, the cdf of the degenerate distribution is the Heaviside step function about the mean:

$$\mathrm{F}(x) = \begin{cases} 1, \text{ for } x \geq \text{mean} \\ 0, \text{ otherwise} \end{cases}$$

Here, "mean" is denoted as the single, constant value represented by the distribution.

The described implementations in the following sections do not apply when variance is set to zero. An implementation of the probabilistic description explained above is invoked instead in this case.

Gaussian distribution

Computation of the pdf

The following straightforward formulation of the Gaussian pdf , defined for $x, \mu \in \Re$ and $\sigma^2 > 0$, is vulnerable to a variety of premature overflow and underflow issues, including premature overflow by the exponent term

$-\frac{(x-\mu)^2}{2\sigma^2}$, which would, in turn, cause premature underflow of the exponentiated expression, $\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$:

$$\mathrm{pdf}(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where μ and σ are the mean and standard deviation of the Gaussian distribution, respectively.

A simple reformulation, however, eliminates the risk of premature overflow and underflow:

$$\begin{aligned}\mathrm{pdf}(x)&=\exp(\ln(\mathrm{pdf}(x)))=\exp\left(\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)\right)\\&=\exp\left[-\frac{1}{2}\left((x-\mu)\left(\frac{x-\mu}{\mathrm{var}}\right)+\ln(\mathrm{var})+\ln(2\pi)\right)\right]\end{aligned}$$

where the term $(x-\mu)\left(\frac{x-\mu}{\mathrm{var}}\right)$ overflows only when the true value of the pdf is zero, even when premature overflow occurs by the terms $(x-\mu)$ or $\left(\frac{x-\mu}{\mathrm{var}}\right)$.

Underflow of the term $(x-\mu)\left(\frac{x-\mu}{\mathrm{var}}\right)$ is not an issue since:

$$\exp\left[-\frac{1}{2}(a\pm\min)\right]=1 \text{ for } a\in\{F: a\neq a\pm\min\}$$

where F is the set of all numbers representable in binary double (64-bit) floating-point precision and a is set as the expression 1n(var)+1n(2π).

Computation of the cdf

Where the error function is numerically stable for any real number input and has the range [-1,1], the following formula for the Gaussian cdf is accurate only when computation of the term $\frac{x-\mu}{\sigma\sqrt{2}}$ is numerically stable:

$$\mathrm{cdf}(x)=\frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

A numerically stable calculation of the term $\frac{x-\mu}{\sigma\sqrt{2}}$ is done first by changing the order of operations:

$$\frac{x-\mu}{\sigma\sqrt{2}} = \frac{\left[\frac{(x-\mu)}{\sqrt{2}}\right]}{\sigma}.$$

When the term $x-\mu$ overflows, the term $\left|\frac{x-\mu}{\sigma\sqrt{2}}\right|$ is computed as follows:

$$\left|\frac{x-\mu}{\sigma\sqrt{2}}\right| = \exp\left(\ln\left|\frac{x-\mu}{\sigma\sqrt{2}}\right|\right) = \exp\left(\ln|x-\mu| - \ln(\sigma) - \ln\left(\sqrt{2}\right)\right) = \exp\left(\ln(2) + \ln\left|\frac{x}{2} - \frac{\mu}{2}\right| - \ln(\sigma) - \frac{\ln(2)}{2}\right)$$

where max = highest representable number in binary double (64-bit) floating-point precision (approximately 1.8×10^{308}).

The absolute value is computed to avoid the potential, impossible computation of a negative logarithm. The negative of the computed term $\left|\frac{x-\mu}{\sigma\sqrt{2}}\right|$ is inputted into the error function for $\frac{x-\mu}{\sigma\sqrt{2}} < 0 \Leftrightarrow x-\mu < 0$.

The term $\frac{(x-\mu)}{\sqrt{2}}$ cannot underflow for all representable x and μ since:

$$\frac{\min}{\sqrt{2}} = \min$$

in binary double (64-bit) floating-point precision.

Computing the rearrangement for when the term $|x-\mu|$ overflows allows the final value of the term $\left|\frac{x-\mu}{\sigma\sqrt{2}}\right|$ to compute to as low as $\frac{1}{\sqrt{2}}$ rather than returning ∞, depending on the value of the term σ in the denominator.

A numerically stable algorithm of the error function for any real number input and with a range [-1,1] computes the regularized incomplete gamma function $P(s,x) = \frac{\gamma(s,x)}{\Gamma(s)} = \frac{1}{\Gamma(s)}\int_0^x r^{s-1}e^{-r}\,dr$, using either the series or continued fraction representation:

$$\operatorname{erf}(t) = \begin{cases} -P\left(\frac{1}{2}, t^2\right), \text{ for } t < 0 \\ P\left(\frac{1}{2}, t^2\right), \text{ otherwise} \end{cases}$$

Since the algorithm for computing the lower incomplete gamma function is only stable for finite representable inputs, where $t^2 \le \max$, which already computes to 1 at $t^2 = 10$, the error function is set to -1 and +1 for $t \le -\sqrt{10}$ and $t \ge \sqrt{10}$, respectively.

Lognormal distribution

Conversion of mean and variance to/from μ and σ

Given mean and variance, it is possible to compute the unique parameters for the lognormal distribution μ and σ by the following formulae:

$$\mu = \ln(\text{mean}) - \frac{1}{2}\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right)$$
$$\sigma = \sqrt{\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right)}$$

However, both of these expressions are susceptible to premature overflow or underflow by the term $\frac{\text{var}}{\text{mean}^2}$. The formulas may be made numerically stable by rearrangement.

Firstly, rearranging the term $\frac{\text{var}}{\text{mean}^2}$ to the term $\frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)$ eliminates the potential of premature underflow or overflow of mean^2, thereby modifying the expression for μ and σ to the following:

$$\mu_1 = \ln(\text{mean}) - \frac{1}{2}\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right) = \ln(\text{mean}) - \frac{1}{2}\ln\left(1 + \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)\right)$$

$$\sigma_1 = \sqrt{\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right)} = \sqrt{\ln\left(1 + \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)\right)}$$

In the case where the term $\frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)$ overflows the expression for μ and σ reduces to the following computable forms in binary double (64-bit) floating-point precision:

$$\mu = \ln(\text{mean}) - \frac{1}{2}\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right) \xrightarrow{\text{for } \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right) > \max} \mu_2 = \ln(\text{mean}) - \frac{1}{2}\ln\left(\frac{\text{var}}{\text{mean}^2}\right) = 2\ln(\text{mean}) - \frac{1}{2}\ln(\text{var})$$

$$\sigma = \sqrt{\ln\left(1 + \frac{\text{var}}{\text{mean}^2}\right)} \xrightarrow{\text{for } \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right) > \max} \sigma_2 = \sqrt{\ln\left(\frac{\text{var}}{\text{mean}^2}\right)} = \sqrt{\ln(\text{var}) - 2\ln(\text{mean})}$$

Assuming round-off errors are negligible, premature overflow is eliminated when calculating μ by computing μ_2 when the term $\frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)$ overflows and computing μ_1 otherwise:

$$\mu = \begin{cases} \mu_2, \text{ for } \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right) > \max \\ \mu_1, \text{ otherwise} \end{cases}$$

Similarly, a numerically stable algorithm for calculating σ computes σ_2 when the term $\frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)$ overflows and computes σ_1 otherwise:

$$\sigma = \begin{cases} \sigma_2, \text{ for } \frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right) > \max \\ \sigma_1, \text{ otherwise} \end{cases}$$

Underflow of the term $\frac{1}{\text{mean}}\left(\frac{\text{var}}{\text{mean}}\right)$ can be ignored when computing μ and σ since:

$$1 + a = 1$$

for $a \le \min$ in binary double (64-bit) floating-point precision, where min = smallest representable number in binary double (64-bit) floating-point precision (approximately 4.9×10^{-324}).

Conversely, it is possible to compute mean and variance, given μ and σ, by the following formulae:

$$\text{mean} = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$
$$\text{var} = \left(\exp\left(\sigma^2\right) - 1\right)\exp\left(2\mu + \sigma^2\right)$$

A numerically stable algorithm for performing these operations compute different expressions that are manipulated yet equivalent for different combinations of μ and σ.

To stably compute the mean, only a simple change in the order of operations is necessary:

$$\text{mean} = \exp\left(\mu + \frac{\sigma^2}{2}\right) = \exp\left(\mu + \sigma\left(\frac{\sigma}{2}\right)\right)$$

where the term, $\sigma\left(\frac{\sigma}{2}\right)$, overflows only when the true value of the mean overflows.

Two rearranged but mathematically equivalent expressions are also used in different situations to compute variance:

$$\mathrm{var}_1 = \left(\exp\left(\sigma^2\right)-1\right)\exp\left(2\mu+\sigma^2\right) = \exp\left(\ln\left[\left(\exp\left(\sigma^2\right)-1\right)\exp\left(2\mu+\sigma^2\right)\right]\right)$$
$$= \exp\left(\ln\left[1-\frac{1}{\exp\left(\sigma^2\right)}\right]+2\mu+2\sigma\sigma\right)$$

$$\mathrm{var}_2 = \exp\left(\ln\left[1-\frac{1}{\exp\left(\sigma^2\right)}\right]+2\mu+2\sigma\sigma\right) = \exp\left(2\left(\frac{\ln\left[1-\frac{1}{\exp\left(\sigma^2\right)}\right]}{2}+\mu+\sigma^2\right)\right)$$

Firstly, the variance is always computed to zero when $\exp\left(\sigma^2\right)$ is equal to 1. Var_2 is computed whenever 2μ < -max and $2\sigma\sigma$ > max. Var_1 is computed in all other cases:

$$\mathrm{var} = \begin{cases} 0, \text{ for } \exp\left(\sigma^2\right)=1 \\ \mathrm{var}_2, \text{ for } 2\mu < -\max \text{ and } 2\sigma\sigma > \max \\ \mathrm{var}_1, \text{ otherwise} \end{cases}$$

The variance is always greater than the largest, positive representable number in binary double (64-bit) floating-point precision whenever the term σ^2 overflows.

Whenever either 2μ or $2\sigma\sigma$ overflows but not both, $\mathrm{var}_1 = \mathrm{var}_2$ in binary double (64-bit) floating-point precision. Underflow of the term $\sigma\left(\frac{\sigma}{2}\right)$ in mean_1 and the term $2\sigma\sigma$ in var_1 is not an issue since:

$$\exp(x+\min) = \exp(x)$$

for all x in binary double (64-bit) floating-point precision. The term $\ln\left[1-\frac{1}{\left(\exp(\sigma)\right)^{\sigma}}\right]$ could never underflow since:

$$\left|\ln(x)\right| > \min$$

for all $x \neq 1$ in binary double (64-bit) floating-point precision. When $x = 1$, $1\mathrm{n}(x)$ is exactly zero.

Computation of the pdf

As with the Gaussian distribution, the following simple formulation of the lognormal pdf, defined for $x \geq 0$, $\mu \in \Re$, and σ > 0, is also susceptible to premature overflow and underflow issues, including premature overflow by the exponent term $-\frac{(\ln(x)-\mu)^2}{2\sigma^2}$ which would, in turn, cause premature underflow of the exponentiated expression $\exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)$:

$$\mathrm{pdf}(x) = \frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)$$

A simple reformulation, however, eliminates the risk of premature overflow and underflow:

$$\begin{aligned}\mathrm{pdf}_1(x) &= \exp(\ln(\mathrm{pdf}(x))) = \exp\left(\ln\left(\frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)\right)\right)\\ &= \exp\left[-\frac{1}{2}\left(\left(\frac{\ln(x)-\mu}{\sigma}\right)^2 + 2\ln(x) + 2\ln(\sigma) + \ln(2\pi)\right)\right]\end{aligned}$$

where the term $\left(\frac{\ln(x)-\mu}{\sigma}\right)^2$ overflows only when the true value of the pdf is zero.

Underflow of the term $\left(\frac{\ln(x)-\mu}{\sigma}\right)^2$ is not an issue since:

$$\exp\left[-\frac{1}{2}(a \pm \min)\right] = 1 \text{ for } a \in \{F : a \neq a \pm \min\}$$

where F is the set of all numbers representable in binary double (64-bit) floating-point precision and a is set as the expression, $2\ln(x) + 2\ln(\sigma) + \ln(2\pi)$.

Since the above rearrangement always computes NaN for $x = 0$, the pdf of $x = \min$ is computed instead in this case, which may or may not necessarily limit to zero depending on the values of μ and σ. Thus, the final algorithm for computing the pdf is as follows:

$$\mathrm{pdf}(x) = \begin{cases}\mathrm{pdf}(\min), \text{ for } x = 0\\ \mathrm{pdf}_1(x), \text{ otherwise}\end{cases}$$

Computation of the cdf

Where the error function is numerically stable for any real number input and has the range [-1,1], the following formula for the lognormal cdf is accurate only when computation of the term $\frac{\ln(x)-\mu}{\sigma\sqrt{2}}$ is numerically stable:

$$\text{cdf}(x)=\frac{1}{2}\left(1+\text{erf}\left(\frac{\ln(x)-\mu}{\sigma\sqrt{2}}\right)\right)$$

A numerically stable calculation of the term $\frac{\ln(x)-\mu}{\sigma\sqrt{2}}$ is done simply by computing parts of it in a certain order as follows:

$$\frac{\ln(x)-\mu}{\sigma\sqrt{2}}=\frac{1}{\sigma}\left(\frac{\ln(x)-\mu}{\sqrt{2}}\right)$$

Here, the term $\left(\frac{\ln(x)-\mu}{\sqrt{2}}\right)$ can never overflow or underflow by itself for all finite μ and nonzero x in binary double (64-bit) floating-point precision since:

$$\frac{\min}{\sqrt{2}}=\min$$

and

$$\ln(x)\pm\max=\pm\max$$

for all finite x in binary double (64-bit) floating-point precision.

The error function is computed with the same algorithm described for computation of the Gaussian cdf.

Exponential distribution

Parametrization

The exponential distribution is characterized by a single, scale parameter β that is equivalent to the mean. The variance is the square of the mean. Since conversions are one step, no rearrangement is necessary or possible:

$$\text{mean}=\beta$$
$$\text{var}=\beta^2$$

Computation of the pdf

Without computing a rearranged form, the following expression for the exponential pdf, defined for $x \geq 0$ and $\beta > 0$, could result in premature underflow of the exponentiated expression, $\exp\left(-\frac{x}{\beta}\right)$, when $\beta < 1$:

$$\mathrm{pdf}(x) = \frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right)$$

The following algebraically equivalent but rearranged expression is accurate for all x and β:

$$\mathrm{pdf}(x) = \frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right) = \exp\left[\ln\left(\frac{1}{\beta}\exp\left(-\frac{x}{\beta}\right)\right)\right] = \exp\left[-\frac{x}{\beta} - \ln(\beta)\right]$$

where the term $\frac{x}{\beta}$ overflows only when the true value of the pdf is zero.

Underflow of the term $\frac{x}{\beta}$ is not an issue since:

$$\exp[a - \min] = 1 \text{ for } a \in \{F : a \neq a \pm \min\}$$

where F is the set of all numbers representable in binary double (64-bit) floating-point precision and a is set as the expression $-\ln(\beta)$.

Computation of the cdf

The exponential cdf can be stably computed by the following, simple expression for all x and β:

$$\mathrm{cdf}(x) = 1 - \exp\left(-\frac{x}{\beta}\right)$$

Computation of the quantile (cdf^{-1})

Since the cdf expression can be easily solved for x, a closed-form formula for the exponential quantile exists, which is stable for all x and β:

$$x = -\beta\ln(1 - \mathrm{cdf}(x))$$

Gamma distribution

Conversion of mean and variance to/from α and β

Given mean and variance, it is possible to compute the unique parameters for the gamma distribution, α and β, by the following formulae:

$$\alpha = \frac{\text{mean}^2}{\text{var}}$$
$$\beta = \frac{\text{var}}{\text{mean}}$$

While the expression for β is stable for all mean and variance, α should be computed as follows:

$$\alpha = \begin{cases} \dfrac{\text{mean}^2}{\text{var}}, \text{for } \dfrac{\text{mean}}{\text{var}} > \max \\ \left(\dfrac{\text{mean}}{\text{var}}\right)\text{mean, otherwise} \end{cases}$$

The converse operation computes mean and variances of the gamma distribution from α and β, by the following formulae:

$$\text{mean} = \alpha\beta$$
$$\text{var} = \alpha\beta^2$$

The expression for mean is stable for all α and β, but variance should be computed as follows:

$$\text{var} = \alpha\beta^2 = \beta(\alpha\beta)$$

Computation of pdf

The following expression for the gamma pdf, defined for $x \geq 0$, $\alpha > 0$, and $\beta > 0$, must be formulated carefully to prevent inaccuracies from numerous potential premature overflow and underflow issues:

$$\text{pdf}(x) = \frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}} \exp\left(-\frac{x}{\beta}\right)$$

But, to obtain a stable rearrangement of the gamma pdf, it is first necessary to notice how the term $\Gamma(\alpha)$ is computed using the following expression for Lanczos approximation of the gamma function (Lanczos 1964):

$$\Gamma(\alpha+1)=\sqrt{2\pi}\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\frac{1}{2}\right)\right]A_\gamma(\alpha)$$

with

$$A_\gamma(\alpha)=\frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)\frac{\alpha}{\alpha+1}+\rho_2(\gamma)\frac{\alpha(\alpha-1)}{(\alpha+1)(\alpha+2)}+\cdots$$

and

$$\rho_k(\gamma)=\sum_{\delta=0}^{k}C_{2\delta}^{2k}\,\frac{\sqrt{2}}{\pi}\left(\delta+\gamma+\frac{1}{2}\right)^{-\left(\delta+\frac{1}{2}\right)}\exp\left(\delta+\gamma+\frac{1}{2}\right)\left(\delta-\frac{1}{2}\right)!$$

where γ is an arbitrarily chosen constant such that $\Re\left(\alpha+\gamma+\frac{1}{2}\right)>0$ so that the infinite series, $A_\gamma(\alpha)$, converges as required and C_j^i denotes the coefficient of the j-th-degree term in the i-th-degree Chebyshev polynomial of the first kind.

Since $\Gamma(\alpha+1)=\alpha\Gamma(\alpha)$, we have:

$$\Gamma(\alpha)=\frac{\Gamma(\alpha+1)}{\alpha}=\frac{\sqrt{2\pi}}{\alpha}\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\frac{1}{2}\right)\right]A_\gamma(\alpha).$$

Since the series, $A_\gamma(\alpha)$, is convergent, it may be truncated to obtain an approximation with desired precision. For fixed γ and truncation order N, computational efficiency is enhanced by resolving the rational fractions in the series $A_\gamma(\alpha)$ into their constituent partial fractions so that:

$$A_\gamma(\alpha)=\frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)\frac{\alpha}{\alpha+1}+\rho_2(\gamma)\frac{\alpha(\alpha-1)}{(\alpha+1)(\alpha+2)}+\cdots\approx b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}$$

where the coefficients $b(\gamma)$ are pre-computed for fixed γ, which are related to the original coefficients $\rho(\gamma)$ of the series $A_\gamma(\alpha)$ and similarly independent of α.

To illustrate how the coefficients $b(\gamma)$ are pre-computed after decomposing rational fractions of the series $A_\gamma(\alpha)$ the process is shown for a simpler case, when the truncation order is 3. First the following is obtained by combining like terms after partial-fraction decomposition of the rational fractions in the truncated series:

$$
\begin{aligned}
A_\gamma(\gamma) &\approx \frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)\frac{\alpha}{\alpha+1}+\rho_2(\gamma)\frac{\alpha(\alpha-1)}{(\alpha+1)(\alpha+2)}+\rho_3(\gamma)\frac{\alpha(\alpha-1)(\alpha-2)}{(\alpha+1)(\alpha+2)(\alpha+3)} \\
&= \frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)\left(1-\frac{1}{\alpha+1}\right)+\rho_2(\gamma)\left(1+\frac{2}{\alpha+1}-\frac{6}{\alpha+2}\right)+\rho_3(\gamma)\left(1-\frac{3}{\alpha+1}+\frac{24}{\alpha+2}-\frac{30}{\alpha+3}\right) \\
&= \frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)+\rho_2(\gamma)+\rho_3(\gamma)+\frac{-\rho_1(\gamma)+2\rho_2(\gamma)-3\rho_3(\gamma)}{\alpha+1}+\frac{-6\rho_2(\gamma)+24\rho_3(\gamma)}{\alpha+2}+\frac{-30\rho_3(\gamma)}{\alpha+3}.
\end{aligned}
$$

Then we notice that:

$$
\begin{aligned}
A_3(\gamma) &\approx \frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)+\rho_2(\gamma)+\rho_3(\gamma)+\frac{-\rho_1(\gamma)+2\rho_2(\gamma)-3\rho_3(\gamma)}{\alpha+1}+\frac{-6\rho_2(\gamma)+24\rho_3(\gamma)}{\alpha+2}+\frac{-30\rho_3(\gamma)}{\alpha+3} \\
&= b_0(\gamma)+\frac{b_1(\gamma)}{\alpha+1}+\frac{b_2(\gamma)}{\alpha+2}+\frac{b_3(\gamma)}{\alpha+3}=b_0(\gamma)+\sum_{n=1}^{3}\frac{b_n(\gamma)}{\alpha+n}
\end{aligned}
$$

when we let:

$$
\begin{aligned}
b_0(\gamma) &= \frac{1}{2}\rho_0(\gamma)+\rho_1(\gamma)+\rho_2(\gamma)+\rho_3(\gamma) \\
b_1(\gamma) &= -\rho_1(\gamma)+2\rho_2(\gamma)-3\rho_3(\gamma) \\
b_2(\gamma) &= -6\rho_2(\gamma)+24\rho_3(\gamma) \\
b_3(\gamma) &= -30\rho_3(\gamma).
\end{aligned}
$$

After examining the dependence of relative error as a function of γ and the truncation order of the series, $A_\gamma(\alpha)$, it has been determined that using γ = 10.900511 and a truncation order of 10 guarantees 16-digit floating-point accuracy required for implementation in binary double (64-bit) floating-point precision, which yields the following coefficients (Pugh 2004):

$$
\begin{aligned}
b_0 &= 2.48574089138753565546\times10^{-5} \\
b_1 &= 1.05142378581721974210 \\
b_2 &= -3.45687097222016235469 \\
b_3 &= 4.51227709466894823700 \\
b_4 &= -2.98285225323576655721 \\
b_5 &= 1.05639711577126713077 \\
b_6 &= -1.95428773191645869583\times10^{-1} \\
b_7 &= 1.70970543404441224307\times10^{-2} \\
b_8 &= -5.71926117404305781283\times10^{-4} \\
b_9 &= 4.63399473359905636708\times10^{-6} \\
b_{10} &= -2.71994908488607703910\times10^{-9}
\end{aligned}
$$

Currently, EASEE uses an implementation of the Lanczos approximation by M. T. Flanagan in his Java Scientific and Numerical Library with γ =

5.0, a truncation order of 6, and an error bound less than 2×10^{-10}, which has the following coefficients:

$$\begin{aligned}
b_0 &= 1.000000000190015\\
b_1 &= 76.18009172947146\\
b_2 &= \text{-}86.50532032941677\\
b_3 &= 24.01409824083091\\
b_4 &= \text{-}1.231739572450155\\
b_5 &= 0.1208650973866179\times10^{\text{-}2}\\
b_6 &= \text{-}0.5395239384953\times10^{\text{-}5}
\end{aligned}$$

Now return to the original, problematic formulation of the gamma pdf:

$$\mathrm{pdf}(x)=\frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}}\exp\left(-\frac{x}{\beta}\right)$$

and set:

$$\begin{aligned}
\Gamma(\alpha)&=\frac{\sqrt{2\pi}}{\alpha}\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\frac{1}{2}\right)\right]A_{\gamma}(\alpha)\\
&\approx\frac{\sqrt{2\pi}}{\alpha}\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\frac{1}{2}\right)\right]\left(b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)
\end{aligned}$$

to obtain:

$$\mathrm{pdf}(x)=\frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}}\exp\left(-\frac{x}{\beta}\right)\approx\frac{x^{\alpha-1}\exp\left(-\dfrac{x}{\beta}\right)}{\left[\dfrac{\sqrt{2\pi}}{\alpha}\left(\alpha+\gamma+\dfrac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\dfrac{1}{2}\right)\right]\left(b_0(\gamma)+\displaystyle\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)\right]\beta^{\alpha}}$$

which is the actual, full expression of the approximated gamma pdf.

This formulation of pdf is unstable when computed within binary double (64-bit) floating-point precision, mostly due to premature overflow by the approximation expression for the term $\Gamma(\alpha)$ and the term $x^{\alpha-1}$ as well as premature underflow by the term $\exp\left(-\frac{x}{\beta}\right)$. In fact, there are many (even relatively small) values for x and α that cause $\Gamma(\alpha)$ and $x^{\alpha-1}$ to overflow simultaneously, since both are rapidly increasing functions as α increases, generating NaN due to the computation of ∞/∞, even though the true ratio

of $\frac{x^{\alpha-1}}{\Gamma(\alpha)}$ is a simple, finite number that is easily representable in binary double (64-bit) floating-point precision. NaN may also be returned even when only $x^{\alpha-1}$ overflows, if the term $\exp\left(-\frac{x}{\beta}\right)$ underflows to zero, which results in the indeterminate form, $0\times\infty$, in the numerator. These observations along with many other potential scenarios that could cause inaccurate (or more often unviable) solutions to the gamma pdf make this formulation quite unpredictable and unsatisfactory.

Firstly, a few basic manipulations of the original formula result in the following, more stable form:

$$
\begin{aligned}
\mathrm{pdf}_1(x) &= \frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}}\exp\left(-\frac{x}{\beta}\right) \\
&\approx \frac{x^{\alpha-1}\exp\left(-\frac{x}{\beta}\right)}{\left[\frac{\sqrt{2\pi}}{\alpha}\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left[-\left(\alpha+\gamma+\frac{1}{2}\right)\right]\left(b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)\right]\beta^{\alpha}} \\
&= \frac{\alpha x^{\alpha-1}\exp\left(\alpha+\gamma+\frac{1}{2}\right)}{\beta^{\alpha}\sqrt{2\pi}\left(b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left(\frac{x}{\beta}\right)} \\
&= \exp\left[\ln\left(\frac{\alpha x^{\alpha-1}\exp\left(\alpha+\gamma+\frac{1}{2}\right)}{\beta^{\alpha}\sqrt{2\pi}\left(b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)\left(\alpha+\gamma+\frac{1}{2}\right)^{\alpha+\frac{1}{2}}\exp\left(\frac{x}{\beta}\right)}\right)\right] \\
&= \exp\left[A-\frac{x}{\beta}+\alpha B\right]
\end{aligned}
$$

where $A=\ln(\alpha)-\ln(x)+\alpha+\gamma+\frac{1}{2}-\frac{\ln(2\pi)}{2}-\ln\left(b_0(\gamma)+\sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha+n}\right)-\frac{\ln\left(\alpha+\gamma+\frac{1}{2}\right)}{2}$ and $B=\ln(x)-\ln(\beta)-\ln\left(\alpha+\gamma+\frac{1}{2}\right)$, which is stable for most x, α, and β. When the term $\frac{x}{\beta}$ overflows and $B > 0$, the following is computed instead:

$$
\mathrm{pdf}_2(x)=\frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}}\exp\left(-\frac{x}{\beta}\right)\approx\exp\left[A-\frac{x}{\beta}+\alpha B\right]=\exp\left[\frac{(A\beta-x+(\alpha\beta)B)}{\beta}\right]
$$

To avoid automatically computing a pdf value of +∞ for $x = 0$ using these rearrangements, the pdf of x = min is returned instead as a very close approximation, which would only limit to zero for sufficiently large α.

Thus, the final algorithm for the gamma pdf computation is as follows:

$$\mathrm{pdf}(x) = \begin{cases} \mathrm{pdf}(\min), \text{ for } x = 0 \\ \mathrm{pdf}_2(x), \text{ for } \dfrac{x}{\beta} > \max \text{ and } B > 0 \\ \mathrm{pdf}_1(x), \text{ otherwise} \end{cases}$$

Here, the term A never overflows for all nonzero, finite, positive values of x and α in binary double (64-bit) floating-point precision and $\gamma = 5.0$, as required to prevent premature overflow.

To confirm this, we first notice that the expressions $b_0(\gamma) + \sum_{n=1}^{N} \frac{b_n(\gamma)}{\alpha + n}$ and $\alpha + \gamma + \frac{1}{2}$ are nonzero and finite in binary double (64-bit) floating-point precision in the given domain of x, α, and γ (i.e., $x \geq 0$, $\alpha > 0$, and $\gamma = 5.0$). This is easy to see with the expression $\alpha + \gamma + \frac{1}{2} = \alpha + 5 + \frac{1}{2}$ which can never equal zero because it is the summation of nonzero, positive numbers and can never overflow since:

$$\max + 5 + \frac{1}{2} = \max$$

in binary double (64-bit) floating-point precision. The expression $b_0(\gamma) + \sum_{n=1}^{N} \frac{b_n(\gamma)}{\alpha + n} = b_0(5) + \sum_{n=1}^{N} \frac{b_n(5)}{\alpha + n}$ can never overflow by examination of the $b(5)$ coefficients for truncation order 6 and noting that $\alpha + n \geq 1$ for any positive, nonzero integer, n. It can never equal zero since $b_0(\gamma) + \sum_{n=1}^{N} \frac{b_n(\gamma)}{\alpha + n} = 0$ implies that $\Gamma(\alpha) = 0$, which contradicts the fact that $\Gamma(\alpha) > 0$ for $\alpha > 0$.

Hence, given that:

$$-744.4400719213812 \approx \ln(\min) \leq \ln(a) \leq \ln(\max) \approx 709.782712893384 \text{ for } a \in \{F : a \neq 0\}.$$

where F is the set of all numbers representable in binary double (64-bit) floating-point precision, we have:

$$A = \ln(\alpha) - \ln(x) + \alpha + \gamma + \frac{1}{2} - \frac{\ln(2\pi)}{2} - \ln\left(b_0(\gamma) + \sum_{n=1}^{N}\frac{b_n(\gamma)}{\alpha + n}\right) - \frac{\ln\left(\alpha + \gamma + \frac{1}{2}\right)}{2} \le \alpha + \gamma + \frac{1}{2} + 5 \times |\ln(\min)| \cdot$$

The expression $\alpha + \gamma + \frac{1}{2} + 5 \times |\ln(\min)| = \alpha + 5 + \frac{1}{2} + 5 \times |\ln(\min)|$ can never overflow since:

$$\max + 5 + \frac{1}{2} + 5 \times |\ln(\min)| = \max$$

in binary double (64-bit) floating-point precision.

The term, B, never overflows for all nonzero, finite, positive values of x, α, and β in binary double (64-bit) floating-point precision and $\gamma = 5.0$:

$$|B| = \left|\ln(x) - \ln(\beta) - \ln\left(\alpha + \gamma + \frac{1}{2}\right)\right| \le 3|\ln(\min)| < \max$$

Underflow of A is not an issue since:

$$\frac{\min}{\max} = 0$$

in binary double (64-bit) floating-point precision.

Underflow of any of one or more of the terms A, $\frac{x}{\beta}$, and αB may be ignored since:

$$\exp(a \pm \min) = 1 \text{ for } a \in \{F : a \ne a \pm \min\},$$

$$\exp(a \pm 2\min) = 1 \text{ for } a \in \{F : a \ne a \pm 2\min\}, \text{ and}$$

$$\exp(\pm 3\min) = 1$$

where F is the set of all numbers representable in binary double (64-bit) floating-point precision and a is set as the expression for the non-underflowing term(s).

Computation of the cdf

Like the error function in the Gaussian and lognormal cdfs, the gamma cdf is also related to the regularized incomplete gamma function

$P(s,x)=\frac{\gamma(s,x)}{\Gamma(s)}=\frac{1}{\Gamma(s)}\int_0^x r^{s-1}e^{-r}dr$, which is likewise computed using either the series or continued fraction representation:

$$\mathrm{cdf}(x)=P\left(\alpha,\frac{x}{\beta}\right)=\frac{\gamma\left(\alpha,\frac{x}{\beta}\right)}{\Gamma(\alpha)}$$

While the individual terms $\gamma\left(\alpha,\frac{x}{\beta}\right)$ and $\Gamma(\alpha)$ overflow at modest values of x, α, and β, the regularized incomplete gamma function itself never overflows since it has the limiting values:

$$P(s,0)=0 \text{ and } P(s,\infty)=1.$$

Thus, it is better to implement the exponentiation of the difference of the logarithms of the two terms:

$$\mathrm{cdf}(x)=P\left(\alpha,\frac{x}{\beta}\right)=\frac{\gamma\left(\alpha,\frac{x}{\beta}\right)}{\Gamma(\alpha)}=\exp\left[\ln\left(\frac{\gamma\left(\alpha,\frac{x}{\beta}\right)}{\Gamma(\alpha)}\right)\right]=\exp\left[\ln\left(\gamma\left(\alpha,\frac{x}{\beta}\right)\right)-\ln(\Gamma(\alpha))\right]$$

With this rearrangement, the gamma cdf is computable for a much more expansive domain of x, α, and β, since it is only required that the *logarithm* of the two terms $\gamma\left(\alpha,\frac{x}{\beta}\right)$ and $\Gamma(\alpha)$ not overflow. For the vast majority of cases, this rearranged formulation of the gamma cdf is stable. In the rare instance that the rearrangement proves unstable, the algorithm computes a direct numerical integration of the gamma pdf, which is always stable. Information on the implementation of the Gauss-Legendre quadrature rule for efficiently integrating pdfs is given in the final section of the report on numerical issues.

$X \rightarrow X^2$ transformed Rice-Nakagami distribution

The distributions of the powers of certain acoustic and seismic signals are closely approximated by the Rice-Nakagami model. Specifically, the signal-power distribution resembles the $X \rightarrow X^2$ transformed Rice-Nakagami distribution when the amplitude is Rice-Nakagami distributed, since signal power is the square of signal amplitude. For consistency, all sensors are modeled to work with signal power rather than amplitude, thus the $X \rightarrow X^2$ transformed Rice-Nakagami distribution is encoded. It turns out

that it is also simpler to implement the $X \to X^2$ transformed Rice-Nakagami distribution, whose expressions for computing mean and variance from its unique parameters υ and σ are clean polynomials and, unlike the untransformed version, do not involve the half-degree Laguerre polynomial.

Derivation of the $X \longrightarrow X^2$ transformed Rice-Nakagami distribution

Statistically, the $X \to X^2$ transformed Rice-Nakagami distribution is the distribution of X^2 where X is a random variable with a Rice-Nakagami distribution. A fundamental theorem for random variable transformation states that, if $p_T(t)$ is the value of the probability density of the continuous random variable T at t and the function $x = \Phi(t)$ is differentiable and monotonic for all values within the range of T for which its probability density does not equal 0 (i.e., $p_T(t) \neq 0$), the probability density of X is given by:

$$p_X(x) = \begin{cases} p_T\left[\Phi^{-1}(x)\right] \cdot \left| \dfrac{d\Phi^{-1}(x)}{dx} \right|, \text{ if } \dfrac{d\Phi(t)}{dt} \neq 0 \\ 0, \text{ otherwise} \end{cases}$$

Hence, for the Rice-Nakagami distribution, where $x = \Phi(t) = t^2$, we have:

$$p_X(x) = \begin{cases} p_T\left[\sqrt{x}\right] \cdot \left| \dfrac{1}{2\sqrt{x}} \right|, \text{ if } t \neq 0 \\ 0, \text{ otherwise} \end{cases}$$

where:

$\Phi^{-1}(x) = +\sqrt{x}$ since $t \geq 0$,

$$\frac{d\Phi^{-1}(x)}{dx} = \frac{d\left(\sqrt{x}\right)}{dx} = \frac{1}{2\sqrt{x}}, \text{ and}$$

$$\frac{d\Phi(t)}{dt} = 2t \cdot$$

Thus, for $t \neq 0$:

$$
\begin{aligned}
p_X(x) &= p_T\left[\sqrt{x}\right]\cdot\left|\frac{1}{2\sqrt{x}}\right| \\
&= \left[\frac{\sqrt{x}}{\sigma^2}\exp\left(\frac{-\left(x+\nu^2\right)}{2\sigma^2}\right)I_0\left(\frac{\nu\sqrt{x}}{\sigma^2}\right)\right]\cdot\frac{1}{2\sqrt{x}} \\
&= \frac{1}{2\sigma^2}\exp\left(\frac{-\left(x+\nu^2\right)}{2\sigma^2}\right)I_0\left(\frac{\nu\sqrt{x}}{\sigma^2}\right)
\end{aligned}
$$

where I_0 is the zeroth order modified Bessel function of the first kind and υ and σ are the nonnegative parameters of the Rice-Nakagami distribution.

Parameters of the $X \longrightarrow X^2$ transformed Rice-Nakagami distribution

By definition, the mean of transformed pdf, $p_X(x)$, is the integral of all values within the range of X with respect to (i.e., weighted by) its probability density:

$$
E(X) = \int_{-\infty}^{\infty} x p_X(x)dx
$$

which, for $x = \Phi(t) = t^2$ and $t \neq 0$, is equal to the second raw moment μ_2' of the untransformed pdf $p_T(t)$:

$$
E(X) = \int_{-\infty}^{\infty} x p_X(x)dx = \int_{-\infty}^{\infty} \Phi(t)p_T\left[\Phi^{-1}(x)\right]\cdot\left|\frac{d\Phi^{-1}(x)}{dx}\right|dx = \int_{-\infty}^{\infty}\Phi(t)p_T(t)dt = E(\Phi(T)) = E\left(T^2\right) = \mu_2'
$$

Since $\mu_2' = 2\sigma^2 + \nu^2$ for the Rice-Nakagami distribution, we have:

$$
E(X) = \mu_2' = 2\sigma^2 + \nu^2
$$

Also by definition, the variance of the transformed random variable, X, can be expanded as follows:

$$
\begin{aligned}
Var(X) &= E\left[(X - E(X))^2\right] = E\left[X^2 - 2XE(X) + E(X)^2\right] \\
&= E\left(X^2\right) - 2E(X)E(X) + E(X)^2 = E\left(X^2\right) - 2E(X)^2 + E(X)^2 = E\left(X^2\right) - E(X)^2
\end{aligned}
$$

which, for $x = \Phi(t) = t^2$ and $t \neq 0$, may be expressed using the second and fourth raw moments μ_2' and μ_4' of the untransformed pdf $p_T(t)$:

$$
\begin{aligned}
Var(X) &= E\left(X^2\right) - E(X)^2 = E\left(T^4\right) - E\left(T^2\right)^2 = \mu_4' - \left(\mu_2'\right)^2 = \left(8\sigma^4 + 8\sigma^2\nu^2 + \nu^4\right) - \left(2\sigma^2 + \nu^2\right)^2 \\
&= \left(8\sigma^4 + 8\sigma^2\nu^2 + \nu^4\right) - \left(4\sigma^4 + 4\sigma^2\nu^2 + \nu^4\right) = 8\sigma^4 + 8\sigma^2\nu^2 + \nu^4 - 4\sigma^4 - 4\sigma^2\nu^2 - \nu^4 = 4\sigma^4 + 4\sigma^2\nu^2
\end{aligned}
$$

Given $E(X)=2\sigma^2+\nu^2$ and $Var(X)=4\sigma^4+4\sigma^2\nu^2$ for the $X\rightarrow X^2$ transformed Rice-Nakagami distribution, the parameters, σ and υ, of the transformed pdf $p_X(x)$ may be obtained from $E(X)$ and $Var(X)$ by the following formulae:

$$\sigma=\sqrt{\frac{E(X)-\sqrt{E(X)^2-Var(X)}}{2}}$$
$$\nu=\left[E(X)^2-Var(X)\right]^{\frac{1}{4}}$$

Conversion of mean and variance to/from v and σ

From the previous section, we have the following formulae for computing the parameters, σ and υ, from mean and variance for the $X\rightarrow X^2$ transformed Rice-Nakagami distribution:

$$\sigma=\sqrt{\frac{\text{mean}-\sqrt{\text{mean}^2-\text{var}}}{2}}$$
$$\nu=\left[\text{mean}^2-\text{var}\right]^{\frac{1}{4}}$$

Conversely, we have the following expressions for mean and variance with respect to the parameters, σ and υ, for the $X\rightarrow X^2$ transformed Rice-Nakagami distribution:

$$\text{mean}=2\sigma^2+\nu^2$$
$$\text{var}=4\sigma^4+4\sigma^2\nu^2$$

While the exact details are not included here, these conversion expressions are made stable using similar techniques used for the other previously described distributions.

Computation of the pdf

As derived earlier, the $X\rightarrow X^2$ transformed Rice-Nakagami pdf, defined for $x\geq 0$, $\nu\geq 0$, and $\sigma\geq 0$, is as follows:

$$\text{pdf}(x)=\frac{1}{2\sigma^2}\exp\left(\frac{-\left(x+\nu^2\right)}{2\sigma^2}\right)I_0\left(\frac{\nu\sqrt{x}}{\sigma^2}\right)$$

where I_0 is the zeroth order modified Bessel function of the first kind. Again, details on implementing this pdf are not included here. As with the gamma function in the gamma pdf, it is important to examine and dissect the approximating method used to compute I_0 when manipulating the entire pdf expression into stable rearrangements.

Computation of the cdf

To compute the $X \to X^2$ transformed Rice-Nakagami cdf, the transformed pdf is numerically integrated using Gauss-Legendre quadrature. While this approach is generally more computationally intensive than a direct approximation, its stability is guaranteed if the algorithm for the pdf computation is fully stable. More importantly, it is a simpler alternative to devising a stable approximation of the first-order Marcum Q-Function in the $X \to X^2$ transformed Rice-Nakagami cdf. An overview on implementing Gauss-Legendre quadrature to numerically integrate pdfs is in the final section of this appendix.

As with the pdf, it is also possible to derive the $X \to X^2$ transformed Rice-Nakagami cdf. In general, any continuous random variable X that is related to another continuous random variable T by a function $X = \Phi(T)$ has the following cdf:

$$F_X(x) = P(X \le x) = P(\Phi(T) \le x) = P(\{t \in \Psi : \Phi(T) \le x\}) = \int_{\{t \in \Psi : \Phi(T) \le x\}} f_T(t)dt$$

where Ψ is the support set of T, F_X is the cdf of X, and f_T is the pdf of T.

Given that the function Φ is strictly increasing, as is the case for $X = \Phi(T) = T^2$, we have:

$$F_X(x) = \int_{\{t \in \Psi : \Phi(T) \le x\}} f_T(t)dt = P\left(t \le \Phi^{-1}(x)\right) = \int_{-\infty}^{\Phi^{-1}(x)} f_T(t)dt = F_T\left(\Phi^{-1}(x)\right)$$

where F_T is the cdf of T.

Since the untransformed Rice-Nakagami distribution has the following cdf:

$$\mathrm{cdf}(x) = 1 - Q_1\left(\frac{\nu}{\sigma}, \frac{x}{\sigma}\right)$$

we have the following expression for the $X \to X^2$ transformed Rice-Nakagami cdf:

$$F_X(x) = F_T\left(\Phi^{-1}(x)\right) = 1 - Q_1\left(\frac{\nu}{\sigma}, \frac{\Phi^{-1}(x)}{\sigma}\right) = 1 - Q_1\left(\frac{\nu}{\sigma}, \frac{\sqrt{x}}{\sigma}\right)$$

where $\Phi^{-1}(x) = +\sqrt{x}$, since $\Psi = \{t \in \Re : t \geq 0\}$ and Q_1 is the first-order Marcum Q-Function. A numerical integration of the pdf is favored in this case to avoid formulating stable implementations of Q_1, the first-order Marcum Q-Function.

Stable implementation of Gauss-Legendre quadrature on arbitrary integration intervals

Given the abscissas $\{x_{N,k}\}_{k=1}^{N}$ and weights $\{w_{N,k}\}_{k=1}^{N}$ for the N-point Gauss-Legendre rule over [-1,1], it is possible to apply the rule on a function $f(t)$ over an arbitrary interval $[a,b]$ by the following change of variable:

$$t = \frac{a+b}{2} + \frac{b-a}{2}x \text{ and } dt = \frac{b-a}{2}dx$$

Then the following relationship:

$$\int_a^b f(t)dt = \int_{-1}^{1} f\left(\frac{a+b}{2} + \frac{b-a}{2}x\right)\frac{b-a}{2}dx$$

is used to obtain this N-point Gauss-Legendre quadrature formula:

$$\int_a^b f(t)dt = \frac{b-a}{2}\sum_{k=1}^{N} w_{N,k}\, f\left(\frac{a+b}{2} + \frac{b-a}{2}x_{N,k}\right).$$

For the technique to stably integrate a function $f(t)$ over the interval $[a,b]$, it is only required that the abovementioned change-of-variable operation is made stable, provided that the algorithm for the N-point Gauss-Legendre rule stably computes all the abscissas and weights over [-1,1] for some specified N and that the function $f(t)$ is stable over the specified interval $[a,b]$. For integrating pdfs with nonnegative support, the expression $(b - a) / 2$ is always stable. However, whenever the expression $(a + b) / 2$ overflows, it should be computed using the distributed formula $x / 2 + y / 2$ (but *only* then). Thus, we have for all pdf with nonnegative support:

$$\text{cdf}(b) = \int_a^b \text{pdf}(t)dt = \begin{cases} \dfrac{(b-a)}{2}\displaystyle\sum_{k=1}^{N} w_{N,k}\, f\left(\dfrac{a}{2} + \dfrac{b}{2} + \dfrac{(b-a)}{2}x_{N,k}\right), \text{for } a+b > \max \\ \dfrac{(b-a)}{2}\displaystyle\sum_{k=1}^{N} w_{N,k}\, f\left(\dfrac{(a+b)}{2} + \dfrac{(b-a)}{2}x_{N,k}\right), \text{otherwise} \end{cases}$$

since $\frac{(a+b)}{2} > \max \Leftrightarrow a + b > \max$.

REPORT DOCUMENTATION PAGE						*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**						
1. REPORT DATE *(DD-MM-YYYY)* 31-12-2010		**2. REPORT TYPE** Final Technical Report				**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Probability and Statistics in Sensor Performance Modeling						**5a. CONTRACT NUMBER**
						5b. GRANT NUMBER
						5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Kenneth K. Yamamoto, D. Keith Wilson, and Chris L. Pettit						**5d. PROJECT NUMBER**
						5e. TASK NUMBER
						5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center Cold Regions Research and Engineering Laboratory 72 Lyme Road Hanover, NH 03755-1290						**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CRREL TR-10-12
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Headquarters U.S. Army Corps of Engineers Washington, DC 20314-1000						**10. SPONSOR/MONITOR'S ACRONYM(S)**
						11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT Signals from many military targets of interest are often strongly randomized, due to the irregular mechanisms by which the signals are generated and propagated. In particular, complicated and dynamic terrestrial/atmospheric environments (with man-made objects, vegetation, and turbulence) randomize signals through random atmospheric and terrestrial processes affecting the propagation. Signals may also be considered random due to uncertainties in the knowledge of the propagation environment and target-sensor geometry. Predictions of sensor performance and recommendations of sensor types and placements derived from them, thus, should account for the random nature of the sensed signals. This report discusses software-modeling approaches for characterizing signals subject to random generation and propagation mechanisms. By representing signals with random variables, they are manipulated statistically to make probabilistic predictions of sensor performance. Both the theory and implementation in a general, object-oriented software design for battlefield signal transmission and sensing is explained. The Java-language software program is called Environmental Awareness for Sensor and Emitter Employment. Some important numerical issues in the implementation are also discussed.						
15. SUBJECT TERMS battlefield sensors, battlespace signal modeling, decision-support tool, modeling and simulation, object-oriented programming, EASEE, uncertainty, environmental effects, probability, statistics						
16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**		**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	U	62		**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18